(2)

RESEARCH AND DEVELOPMENT TECHNICAL REPORT

SLCET-TR-90-11

DTIC FILE COPY

AD-A231 519

NUMERICAL ANALYSIS OF COMPOSITE DIELECTRICS: PRELIMINARY RESULTS FOR TWO DIMENSIONS

STEPHEN WALLIN (UNIV OF SOUTHERN COLORADO)
JOHN KOSINSKI
ARTHUR BALLATO
ELECTRONICS TECHNOLOGY AND DEVICES LABORATORY

SEPTEMBER 1990

DISTRIBUTION STATEMENT

Approved for public release;
distribution is unlimited.

US ARMY
LABORATORY COMMAND
FORT MONMOUTH, NEW JERSEY 07703-5000

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The citation of trade names and names of manufacturers in this report is not to be construed as official Government indorsement or approval of commercial products or services referenced herein.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188
Exp Date Jun 30, 1986

1a REPORT SECURITY CLASSIFICATION: Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S): SLCET-TR-90-11	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION: US Army Laboratory Command Electronics Tech & Devices Lab; 6b OFFICE SYMBOL (If applicable): SLCET-MA-A	7a NAME OF MONITORING ORGANIZATION
6c ADDRESS (City, State, and ZIP Code): Electronics Technology & Devices Laboratory ATTN: SLCET-MA-A Ft Monmouth, NJ 07703-5000	7b ADDRESS (City, State, and ZIP Code)
8a NAME OF FUNDING/SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
1L162705	AH94	K9	DA303394

11 TITLE (Include Security Classification)

NUMERICAL ANALYSIS OF COMPOSITE DIELECTRICS: PRELIMINARY RESULTS FOR TWO DIMENSIONS (U)

12 PERSONAL AUTHOR(S)

Stephen Wallin (Univ of Southern Colorado), John Kosinski, and Arthur Ballato

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical Report	FROM May 90 TO Aug 90	1990 September	52

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Dielectric materials; composite materials; numerical analysis.
20	03		
11	04		

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The use of composite materials and structures to provide characteristics unattainable directly from the constituent materials is well known. Recently, work has been undertaken to apply this principle to the development of dielectric materials for energy storage applications. It is anticipated that this technology will provide materials for use in capacitors with greater energy density, lower loss, and higher breakdown resistance.

This report describes the results of an effort to analytically model the net or resultant dielectric behavior of such composite dielectric materials. A theoretical model of the composite dielectric is presented and computer code implementing the model is discussed along with preliminary results for the two-dimensional case.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL: John A. Kosinski	22b TELEPHONE (Include Area Code): (908) 544-2843; 22c OFFICE SYMBOL: SLCET-MA-A

CONTENTS

FIGURES

INTRODUCTION

The use of composite materials and structures to provide characteristics unattainable directly from the constituent materials is well known. Perhaps the most widespread example is steel-reinforced concrete for structural applications wherein the high tensile strength of the steel in conjunction with the high compressive strength of the concrete yields a composite material with structural properties far superior to those of either component. More recently, work has been undertaken to apply this principle to the development of dielectric materials for energy storage applications. It is anticipated that this technology will provide materials for use in capacitors with greater energy density, lower loss, and higher breakdown resistance. This report describes the results of an effort to analytically model the net or resultant dielectric behavior of such composite dielectric materials.

The net dielectric behavior of a randomly interspersed composite is dependent on the spatial dimensionality (1-D vs. 2-D vs. 3-D), domain geometries (domain size, domain shape, stratification, etc.), and interconnection effects (percolation). This collective dependence is only partly understood. One averaging law which is useful to the experimentalist because of its relative ease in dealing with multiple components is [1]

$$\epsilon^{\alpha} = \Sigma \, v_k \, \epsilon_k^{\alpha} \qquad (1)$$

where ϵ is the net dielectric permittivity of the composite while ϵ_k is the dielectric permittivity component occupying volume fraction v_k. The exponential factor α depends on the geometry of the constituent components and has been rigorously derived for only the special cases of layers oriented perpendicular to the applied electric field and layers oriented along the direction of the electric field [2]. For the case of layers oriented perpendicular to the applied electric field, $\alpha = -1$ and equation 1 may be interpreted as harmonic averaging. For the case of layers oriented along the direction of the applied electric field, $\alpha = +1$ and equation 1 may be interpreted as arithmetic averaging. Although no obvious physical significance may be immediately attached to the case of $\alpha = 0$, it may be interpreted as geometric averaging. No generalized analytic expression for α exists, and a primary goal of this work is to develop a numerical technique to obtain this factor for different composites.

DIELECTRIC PERMITTIVITY

The dielectric permittivity ϵ relates an applied electric field to the induced displacement field arising as a result of the applied electric field. This relationship is usually written

$$D_i = \epsilon_{ij} \, E_j \qquad (2)$$

where both the displacement D_i and electric field E_j are vector

quantities and the dielectric permittivity ϵ_{ij} is a tensor of rank two over the spatial indices i,j. The summation convention applies to the repeated index j. In the case of isotropic media, equation 2 takes the vector form

$$\mathbf{D} = \epsilon \ \mathbf{E} \qquad (3)$$

with ϵ now written as a scalar quantity. In all practical cases, the dielectric permittivity is a complex quantity and may be denoted

$$\epsilon = \epsilon' + j \ \epsilon'' \qquad (4)$$

wherein $j=\sqrt{-1}$. The Kramers-Kronig relationship, based on time-causality considerations, specifies that the real and imaginary parts of ϵ are not independent. In engineering applications, the complex nature of ϵ is more commonly referred to by the use of the loss tangent, defined as

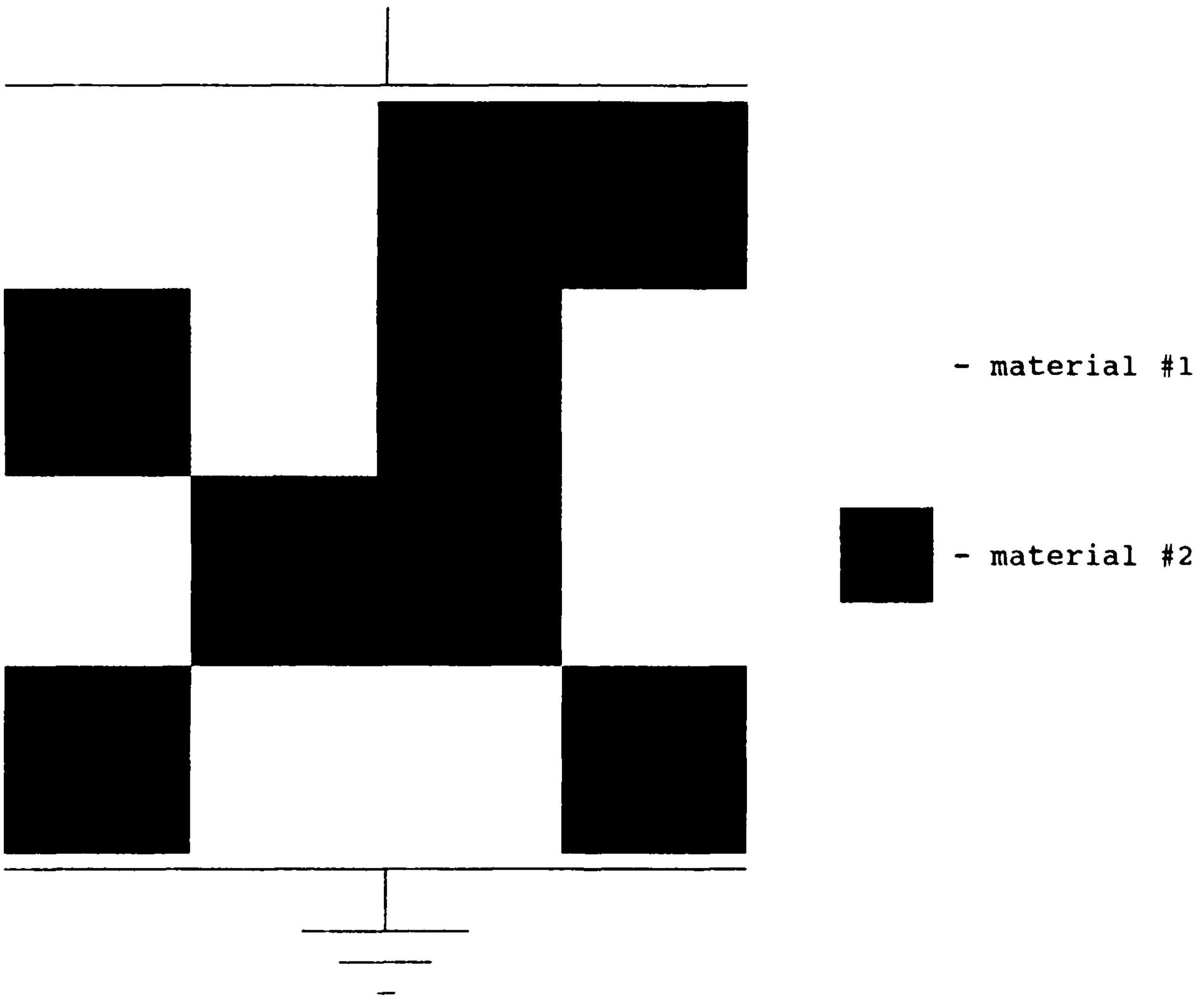


Figure 1. Parallel plate capacitor with composite dielectric.

$$\tan \delta \equiv \epsilon'' / \epsilon' \quad (5)$$

For many common insulators, the loss tangent is on the order of 10^{-4}, and consequently $\tan \delta$ is often approximated by δ. For the purposes of this report, ϵ'' will be taken as zero; the more general case will be treated in a subsequent report.

COMPOSITE DIELECTRIC STRUCTURE

The composite dielectric structure can be modeled as a collection of interconnected domains, with the material properties of each domain being distinct. Figure 1 illustrates a cross-sectional view of a parallel plate capacitor employing a dielectric composed of two distinct materials.

In order to analyze the structure, we superimpose a pixel grid on the specimen. The pixel grid is chosen such that the material within any pixel is homogeneous. This is illustrated in

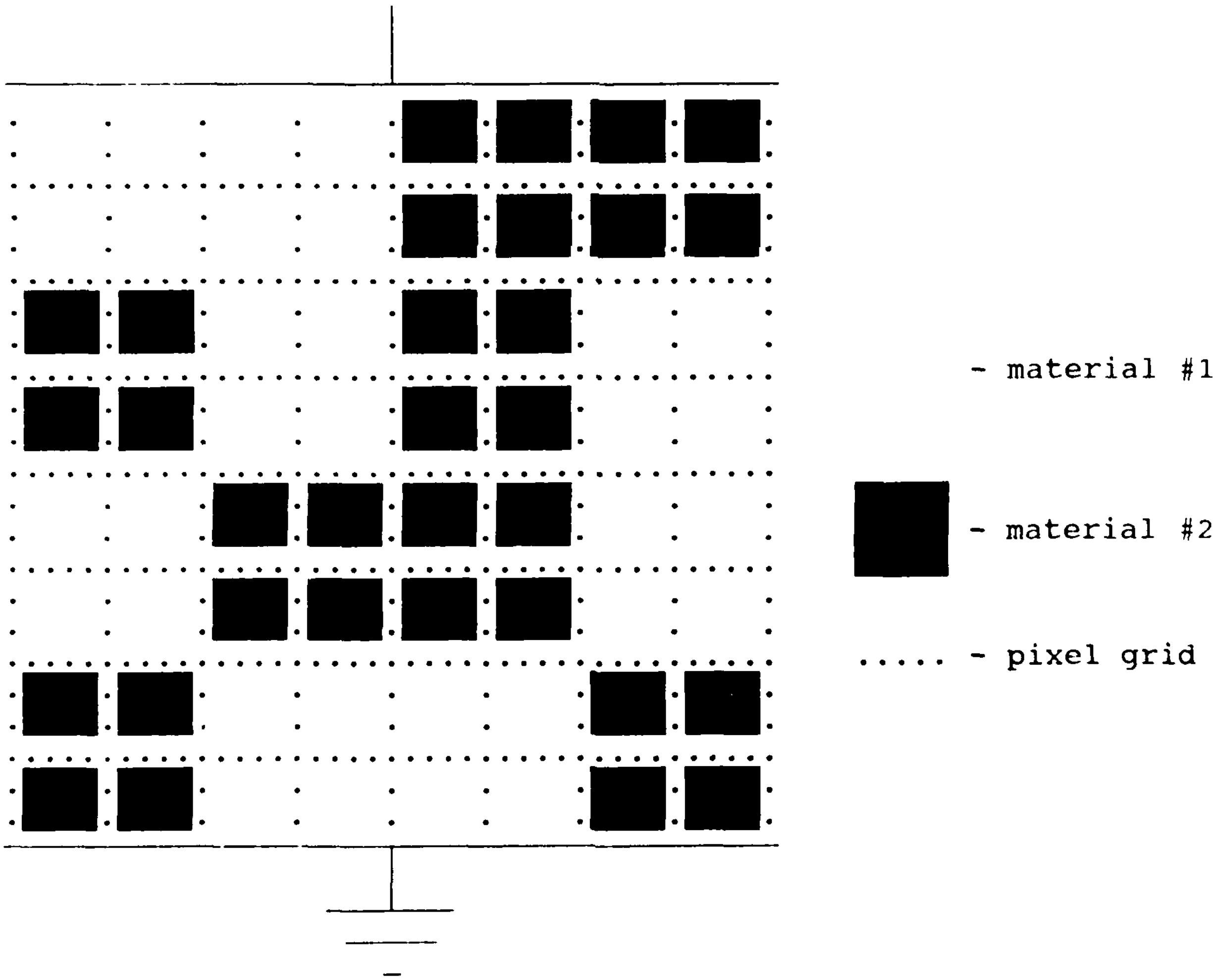


Figure 2. Parallel plate capacitor with composite dielectric showing superimposed pixel grid.

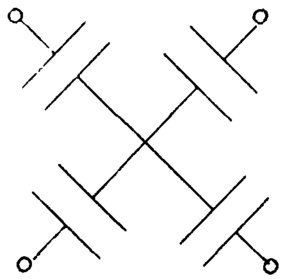


Figure 3. Four-terminal network pixel element model.

Figure 2. The pixel grid forms a Cartesian coordinate system with one axis parallel and one axis perpendicular to the applied electric field. The dielectric properties of the material within any pixel are assigned by means of the pixel grid coordinate system.

We can consider each pixel site as a four-terminal network and can model all nearest-neighbor interactions of any pixel by considering the array of nodes formed at the pixel intersections. This is sufficient to assign a vector displacement to each pixel. The four-terminal network within each pixel is comprised of a set of four capacitors, each connected between a common node at the center of the pixel and one of the corners as illustrated in Figure 3. The value associated with each capacitor is simply the size-normalized capacitance between the pixel center and any corner.

We can redraw the composite dielectric pixel grid of Figure 2 with the four-terminal equivalent network substituted for each pixel element, and obtain the electrical network shown in Figure 4. This network may be analyzed by means of Kirchoff's Laws as applied to the displacement field **D** in the static case or displacement current dD/dt in the dynamic case.

NETWORK ANALYSIS

In the application of Kirchoff's Laws to the analysis of the equivalent electrical network, we are confronted with the choice of using either mesh analysis or nodal analysis. We choose to employ nodal analysis for the following reasons:

1) The numerical solution is more stable using nodal analysis than using mesh analysis. Essentially, as one traverses the composite network, the nodal solution transitions smoothly from pixel to pixel and the relative differences in currents are small, whereas with mesh analysis we may see alternations in the sense of the loop currents which greatly increases the relative differences in currents. From a numerical standpoint, this choice is important in avoiding truncation errors.

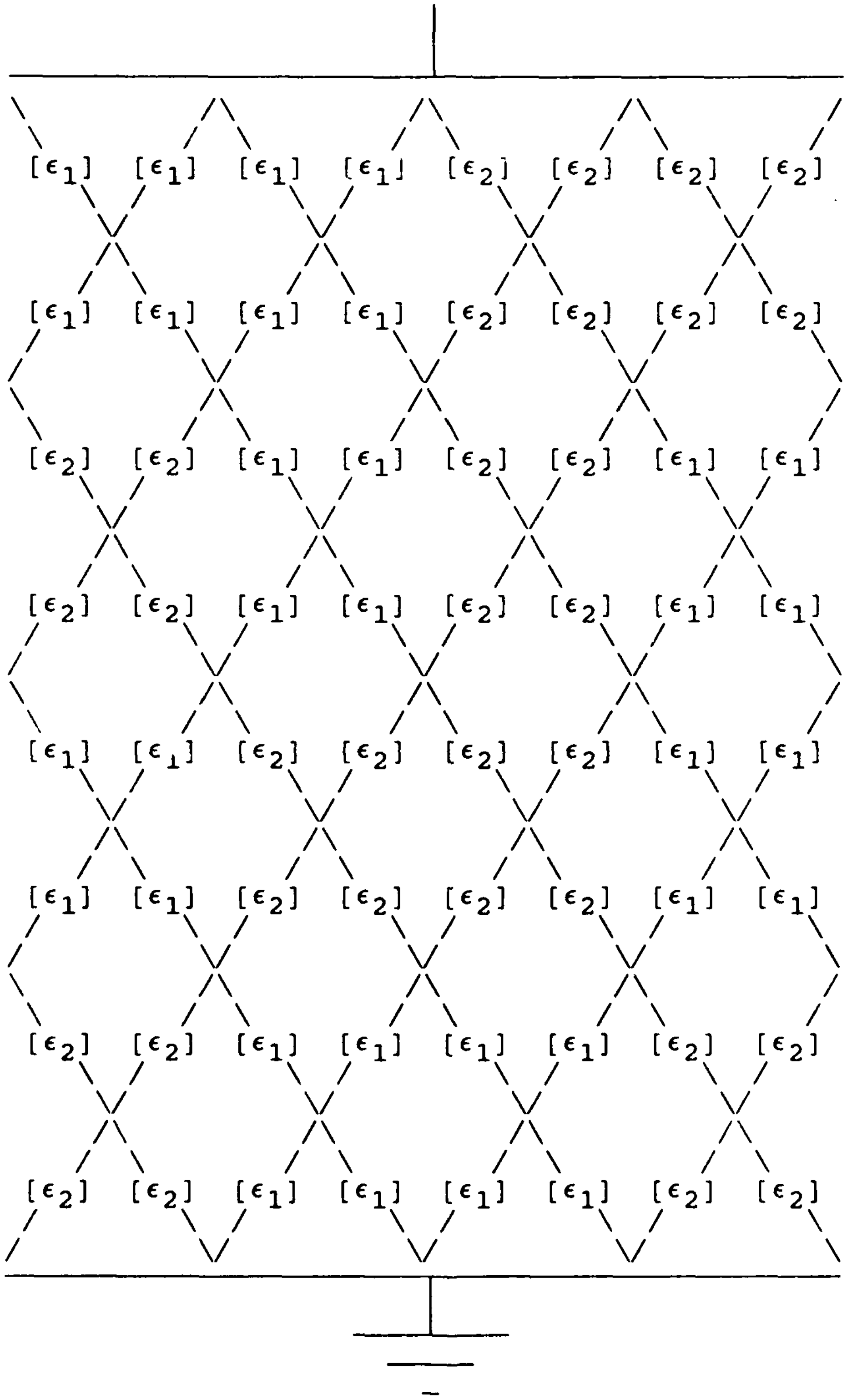


Figure 4. Composite dielectric equivalent electrical circuit where the circuit components have been numbered with subscripts 1 and 2 to correspond with the material components of Figure 2.

2) The boundary conditions are more readily implemented using nodal analysis. This includes both the excitation and lateral boundaries.

Implementation of the nodal analysis results in a succinct description of the network. The currents I_{ij} between the ith node and its nearest neighbors are described by

$$\Sigma I_{ij} = 0 \qquad (5)$$

for all nodes except along the excitation plane. A normalized current is applied to nodes along the excitation plane, for which we may write

$$\Sigma I_{ij} = j w \qquad (6)$$

where jw is the complex radian frequency of the excitation. The summation is taken over the nearest neighbor nodes. The size normalized admittance of the capacitor elements between nodes i and j may be written

$$Y_{ij} = j w \epsilon_{ij} \qquad (7)$$

Equation 7 may be substituted into equations 5 and 6 using Ohm's Law and the potentials at nodes i and j, resulting in

$$\Sigma \epsilon_{ij} (V_i - V_j) = 0 \qquad (8)$$

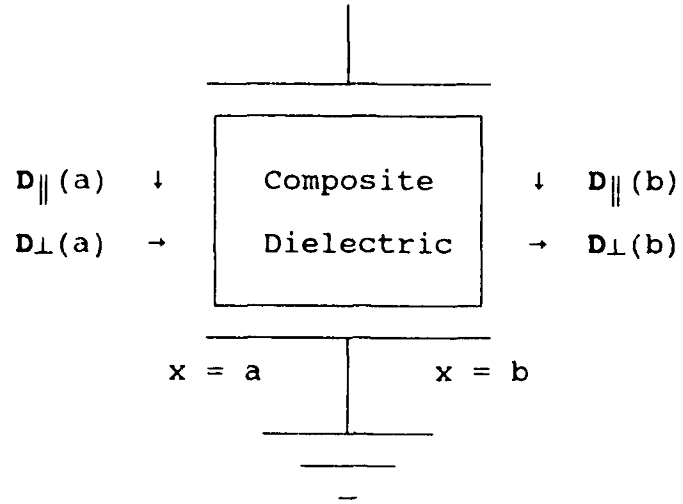
for most nodes and

$$\Sigma \epsilon_{ij} (V_i - V_j) = 1 \qquad (9)$$

for nodes on the excitation plane. The solution of the nodal equations varies with frequency since the ϵ_{ij} of the constituent materials are in reality frequency dependent with the different materials having different frequency dependencies.

BOUNDARY CONDITIONS

In the formulation of the network to be solved, we encounter two common types of boundary conditions concerning the displacement field at the lateral boundaries. The first condition is that of 'insulating' sides, wherein we consider the dielectric specimen to be electrically isolated along the lateral boundaries. In this case, the normal component of the displacement field **D** is considered to be zero at the lateral boundaries. For simplicity, we also assume that fringing electric fields are nonexistent. This boundary condition is applicable to the analysis of isolated samples or lattice cells with mirror-symmetry-plane boundaries. The second condition is that of 'periodic' boundaries, wherein we consider the dielectric specimen to be a repeated cell in a cyclic lattice structure. In this case, the normal component of displacement field is non-zero and the same at both boundaries, as is the tangential component



'Insulating'	'Periodic'
$\mathbf{D}_{\|}(a) \neq \mathbf{D}_{\|}(b) \neq 0$	$\mathbf{D}_{\|}(a) = \mathbf{D}_{\|}(b) \neq 0$
$\mathbf{D}_{\perp}(a) = \mathbf{D}_{\perp}(b) = 0$	$\mathbf{D}_{\perp}(a) = \mathbf{D}_{\perp}(b) \neq 0$

Figure 5. Lateral boundary conditions on the composite dielectric.

of **D**. The two types of boundary conditions are illustrated in Figure 5.

COMPUTER IMPLEMENTATION OF THE NETWORK

The equivalent electrical network representing the composite dielectric may be conveniently solved using computer methods. An important prerequisite to obtaining a solution is the development of an appropriate scheme for identifying both pixel grid nodes and the internodal capacitances. In the scheme adopted here, the pixel grid nodes are numbered sequentially starting from the ground electrode and terminating with the excitation electrode. The internodal capacitances are identified as elements in a two-dimensional array. The numbering scheme is illustrated in Figure 6.

The node numbering scheme is not arbitrary, but is chosen in such a fashion as to maximize the stability of the numerical solution. By beginning at the ground electrode and moving along planes of increasing potential, we can minimize truncation errors. The number of nodes in the pixel grid depends on the lateral boundary conditions. The node numbering schemes for both 'insulating' and 'periodic' boundary conditions are illustrated in Figure 7.

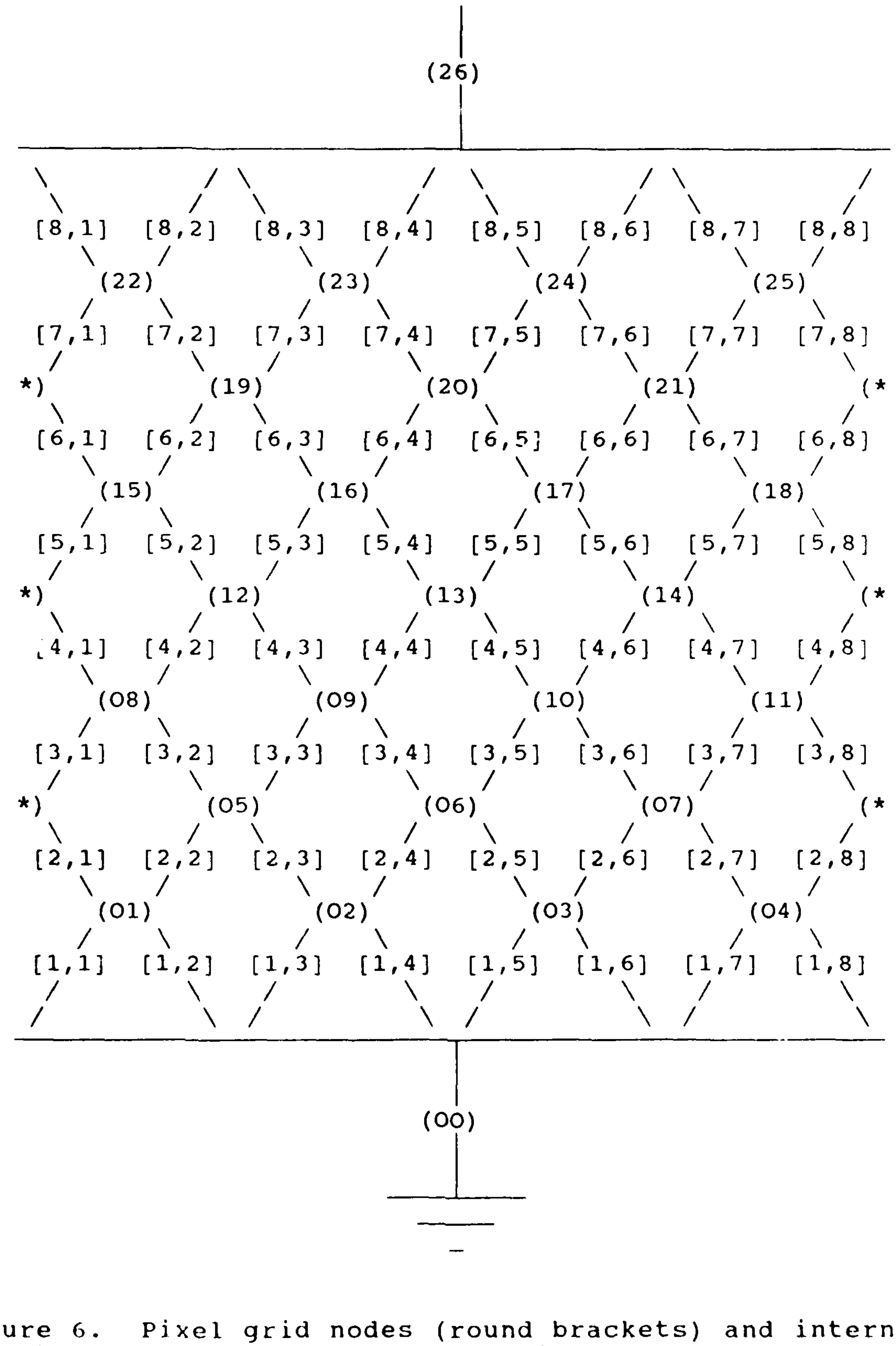
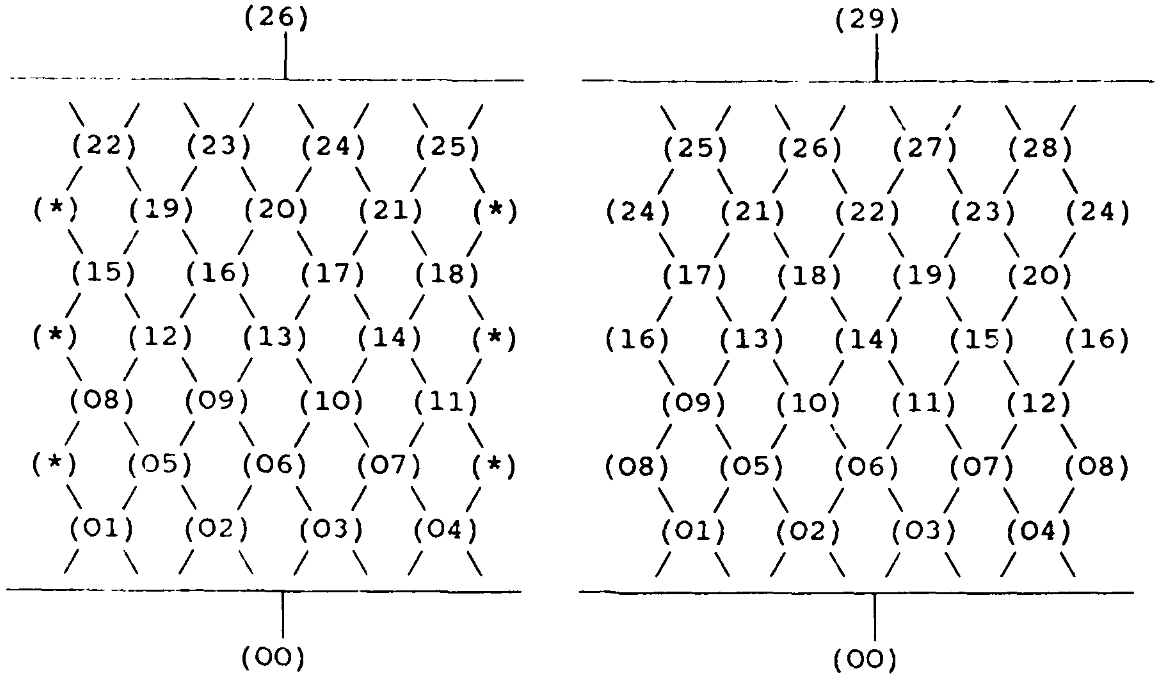


Figure 6. Pixel grid nodes (round brackets) and internodal capacitance (sqaure brackets) numbering scheme for the case of 'insulating' boundary conditions.



(a) (b)

Figure 7. Node numbering scheme in the case of a) 'insulating' and b) 'periodic' boundary conditions. In the case of 'insulating' boundary conditions, the *internodal capacitances* along the starred (*) paths are obtained from series connection of the constituent internodal capacitances. In the case of 'periodic' boundary conditions, the nodes at one boundary effectively 'wrap around' to the other boundary.

INTERACTION MATRIX

Expansion of equations 8 and 9 over the entire pixel grid will yield a set of simultaneous equations which must be solved to obtain the unknown nodal potentials. Once the nodal potentials are known, the net dielectric permittivity is calculated as the quotient of the excitation displacement current and the excitation potential. The full set of equations takes the form

$$\{ I \} = \{ \epsilon \} \{ V \} \qquad (10)$$

where $\{I\}$ is a column vector of the generalized displacement currents, $\{V\}$ is a column vector of the nodal potentials, and $\{\epsilon\}$ is the interaction matrix describing the composite dielectric. The interaction matrix is in general a sparse symmetric matrix, with the sparseness arising from the fact that only nearest-neighbor interaction terms are non-zero. Except for the extrema, there are only five non-zero terms in any row, hence the degree of sparseness increases approximately as the square of the matrix

$$
\begin{bmatrix}
\Sigma & 0 & 0 & 0 & \epsilon & 0 & 0 & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \Sigma & 0 & 0 & \epsilon & 0 & 0 & 0 & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\epsilon & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \epsilon & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \epsilon & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\epsilon & 0 & 0 & 0 & \epsilon & 0 & 0 & \Sigma & 0 & 0 & 0 & \epsilon & 0 & 0 & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \epsilon & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \epsilon & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \epsilon & 0 & 0 & \epsilon & 0 & 0 & 0 & \Sigma & 0 & 0 & \epsilon & 0 & 0 & 0 & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & 0 & 0 & 0 & \epsilon & 0 & 0 & \Sigma & 0 & 0 & 0 & \epsilon & 0 & 0 & \epsilon & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & 0 & 0 & \epsilon & 0 & 0 & 0 & \Sigma & 0 & 0 & \epsilon & 0 & 0 & 0 & \epsilon \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon & \epsilon \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & 0 & 0 & 0 & \epsilon & 0 & 0 & \Sigma & 0 & 0 & \epsilon \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & \epsilon & 0 & 0 & \Sigma & 0 & \epsilon \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & \epsilon & 0 & 0 & \Sigma & \epsilon \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \epsilon & 0 & 0 & \epsilon & \epsilon & \epsilon & \epsilon & \Sigma
\end{bmatrix}
$$

Σ = summation of internodal capacitances

ϵ = internodal capacitance term

Figure 8. Interaction matrix showing the location of non-zero terms using the chosen node numbering scheme.

size. The location of the non-zero elements in the matrix depends upon the node numbering scheme employed, and we have chosen a node numbering scheme such that the interaction matrix is 'banded' with the non-zero terms clustered about the main diagonal. The form of this matrix is illustrated in Figure 8.

Using the 'banded' interaction matrix as shown provides a distinct computational speed and memory size advantage over the use of a 'non-banded' matrix with arbitrary non-zero element locations. The execution speed in the 'banded' case is proportional to the square of the matrix size, whereas in the 'non-banded' case execution speed is proportional to the cube of the matrix size. Memory size is conserved using the 'banded' form since only a small segment of the matrix is operated upon at any time. In this case, the size of the interaction matrix which may be evaluated is limited only by the range of computer addresses available and the time required to perform the calculations.

VERIFICATION OF NUMERICAL SOLUTION

The accuracy of the numerical solution obtained here has been tested in several ways. The first test involved calculation of the exponential averaging factor α for the known cases of stratified layers parallel to the excitation and perpendicular to the excitation. In both cases, for permittivities ϵ_1 and ϵ_2 such that $\epsilon_1/\epsilon_2 < 10^9$ (limited by the computer implementation), the results of the numerical solution were as expected. The second test involved examining the stability of the numerical solution as the grid size was changed for a fixed distribution of constituent materials. No variations in the numerical output were detected as the grid size was varied from 4x4 pixels (26 nodes) to 40x40 pixels (3161 nodes).

The third test involved comparing numerical solutions employing different implementations of node numbering and matrix inversion routines. Essentially, the final version of the numerical analysis code grew out of three earlier, less efficient but computationally accurate implementations. The first code employed a node numbering scheme which started at the center of the dielectric and spiraled outward to the boundaries. This code also employed the matrix inversion routine resident in the programming language. The second version employed the same node numbering scheme but substituted L-U decomposition for matrix inversion. The third version also used the same node numbering scheme, but employed Gaussian elimination in evaluating the interaction matrix. All three versions employed back substitution to verify the numerical solutions. No differences were found between the test case solutions obtained by these versions and the final version.

The fourth and final case involved comparing the current numerical results to those obtained in an earlier work [3]. Again, the current numerical results were in good agreement with the earlier work.

PRELIMINARY RESULTS

The computer code has been used to examine the variation in exponential averaging factor α with the ratio of constituent permittivities in a two-component composite dielectric. Figures 9 through 12 show the results obtained for ϵ_1/ϵ_2 ratios of 1.1, 10, 100, and 1000 respectively. Each figure shows the value of α as a function of constituent volume ratio. Each data point represents the calculated behavior of a capacitor with the indicated volume fractions of constituent dielectrics. For each calculation, the spatial distribution of the constituent dielectrics is determined by a random number generator, the *interaction* matrix is evaluated to determine the net dielectric permittivity, and Equation (1) is solved iteratively for α. One can see from these figures that the behavior of the exponential averaging factor depends on the constituent permittivity ratio and that the behavior falls into three general categories:

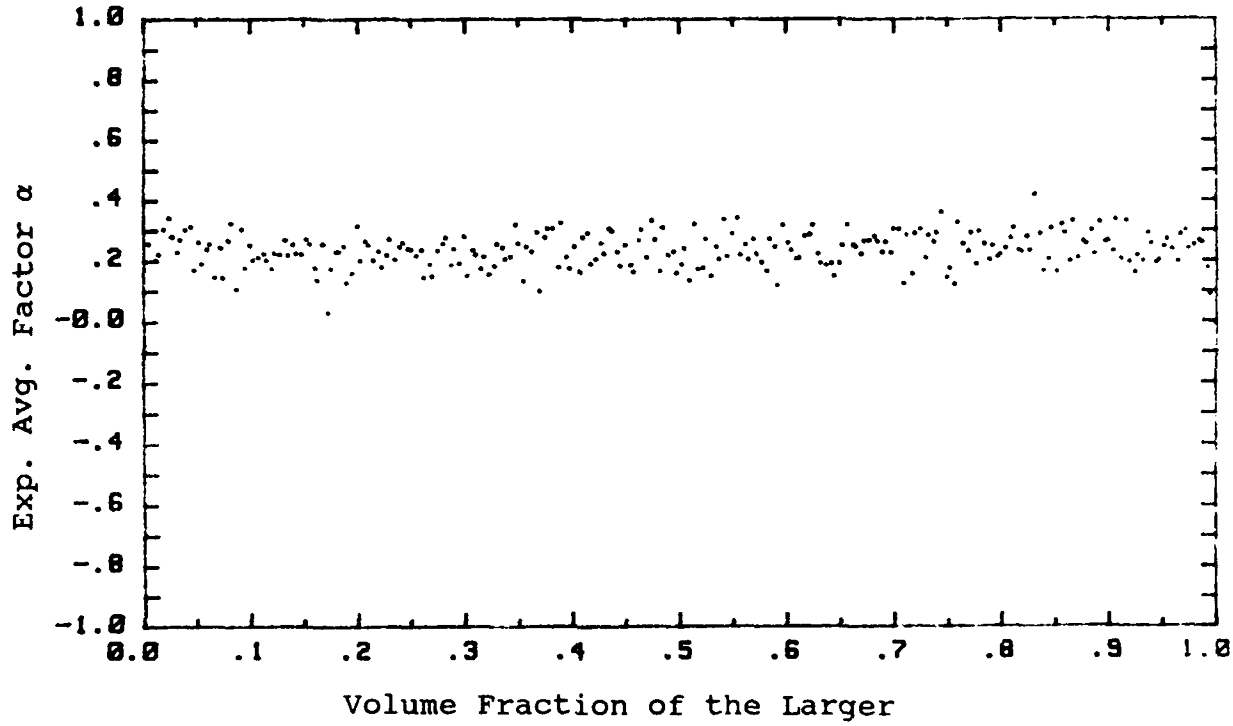


Figure 9. Exponential averaging factor α as a function of constituent volume ratio V_1/V_2 for permittivity ratio $\epsilon_2/\epsilon_1 = 1.1$.

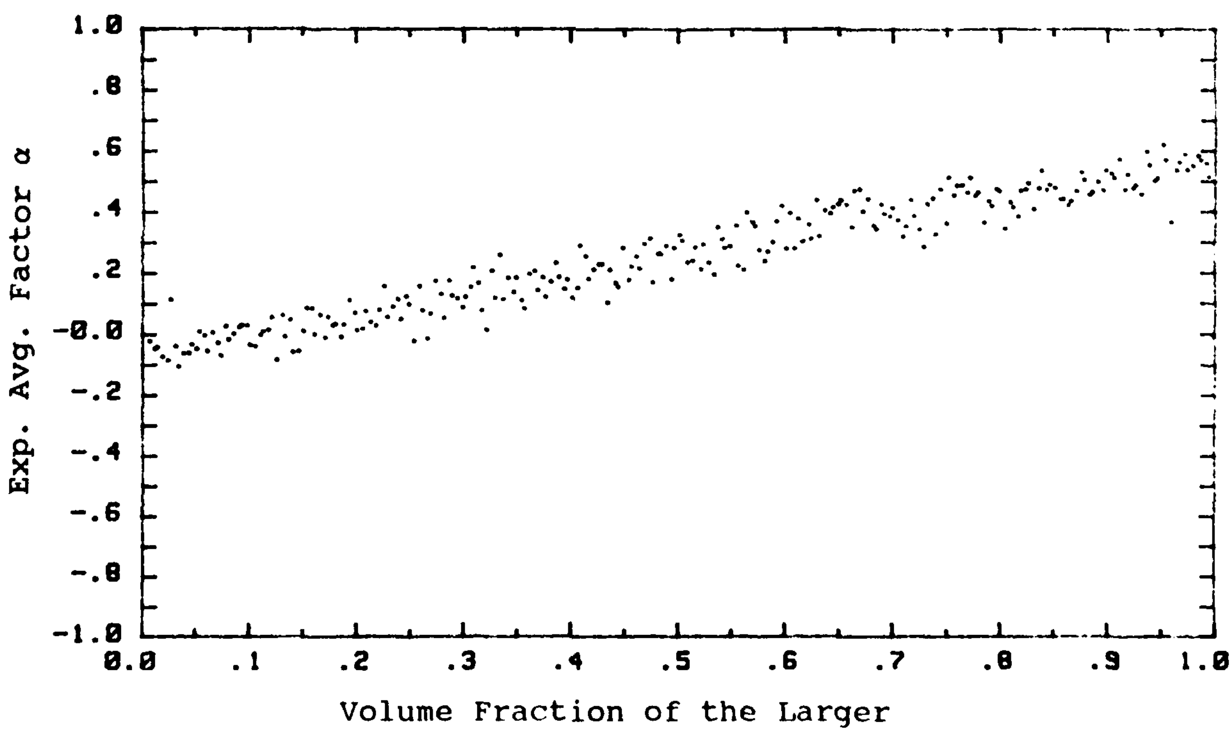


Figure 10. Exponential averaging factor α as a function of constituent volume ratio V_1/V_2 for permittivity ratio $\epsilon_2/\epsilon_1 = 10$.

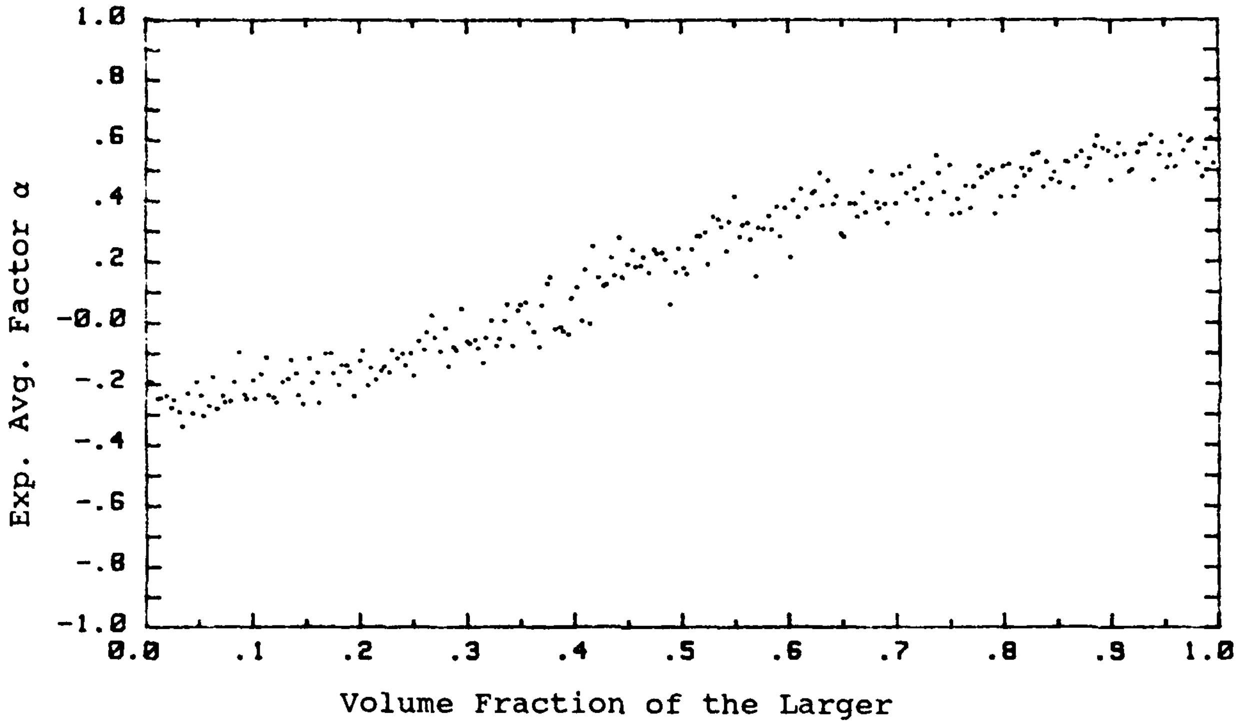


Figure 11. Exponential averaging factor α as a function of constituent volume ratio V_1/V_2 for permittivity ratio $\epsilon_2/\epsilon_1 = 100$.

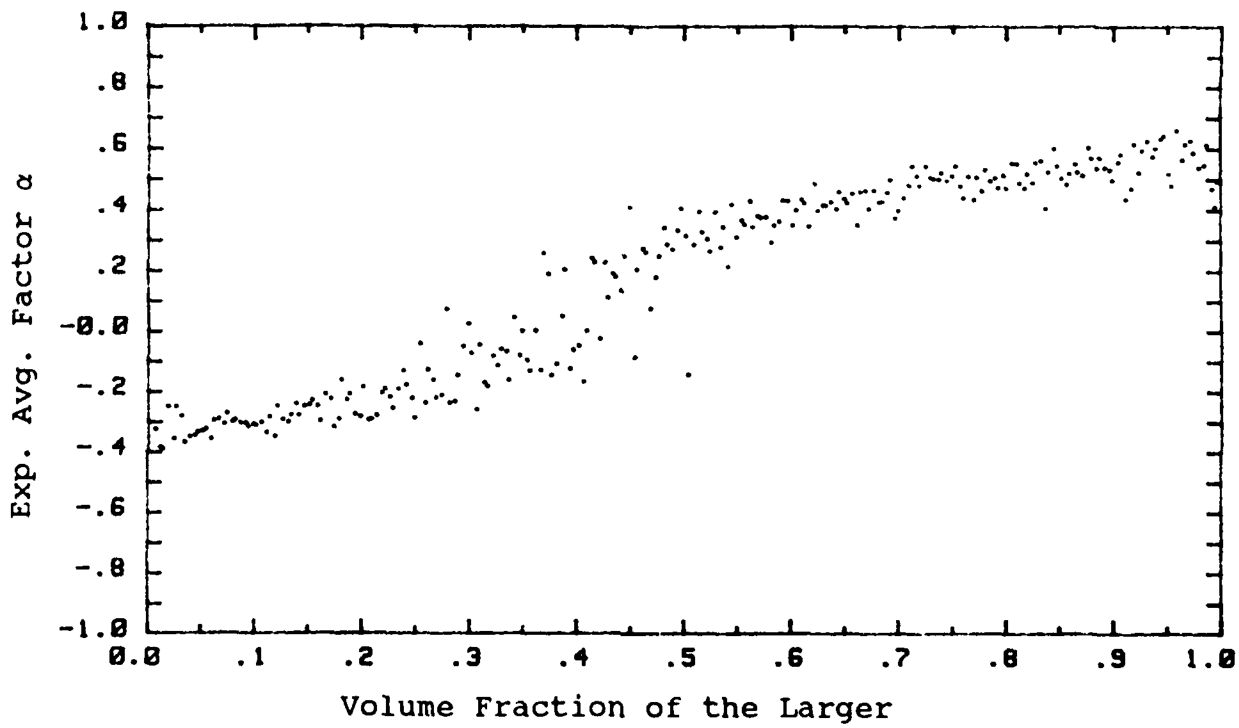


Figure 12. Exponential averaging factor α as a function of constituent volume ratio V_1/V_2 for permittivity ratio $\epsilon_2/\epsilon_1 = 1000$.

a) $\epsilon_1/\epsilon_2 \approx 1$ - the exponential averaging factor α is essentially constant and of value $\approx$0.25.

b) $2 \leq \epsilon_1/\epsilon_2 \leq 100$ - α is approximately linearly dependent on the constituent volume ratio.

c) $\epsilon_1/\epsilon_2 > 100$ - α exhibits a nonlinear constituent volume ratio dependence arising from percolation effects.

Several more cases will need to be examined to determine more accurately the limits of these behavioral categories.

FUTURE WORK

The identification of the exponential averaging factor α as falling into behavioral categories dependent upon the constituent permittivity ratio is of considerable engineering significance. Once the behavioral boundaries are established, it will be possible to develop simple expressions approximating α for the various behavioral categories. Future work in this area should thus include:

a) Evaluation of a sufficient number of test cases to determine the limits of the behavioral categories.

b) Development of engineering tables and approximations for practical determination of α in engineering applications.

c) Extension to the case of complex permittivity, including artificial dielectrics such as dielectric-metal composites.

d) Extension to 3-dimensional sample geometries.

e) Analysis of the nature of percolation effects observed for large constituent permittivity ratios, in particular, analysis of the effects of constituent grain size.

f) Analysis of composites with three or more components.

REFERENCES

[1] K. Lichtenecker and K. Rother, "Die Herleitung des logarithmischen Mishungsgesetzes aus allgemeinen Prinzipien der stationären Strömung," Physikalische Zeitschrift, Vol. 32, pp. 255-260, 1931.

[2] O. Weiner, "Die Theorie des Mischkörpers das Feld der stationaren Strömung," Abhandlungen der Math. - Phys. Klasseder Koenigl., Sachs. Ges. der Wiss., Vol. 32, Leipzig, pp. 509-600, 1912.

[3] S. Wallin, "Dielectric Properties of Heterogeneous Media," Doctoral Dissertation, Department of Physics and Astronomy, University of Wyoming, Laramie, Wyoming, 1985, 148 pp.

Appendix I

Two-Dimensional Numeric Analysis Program

```
10    !  <  <  <  <  <  <  <  "DIEL_LR62D"  >  >  >  >  >  >  >  >
20    ! * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - *
30    !  A main program to evaluate a 2 dimensional composite dielectric
40    !  response for a pixel network of capacitors.
50    !                                            S. R. Wallin, 6/5/90
60    ! * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - *
70    PRINT " MEMORY IS";VAL(SYSTEM$("AVAILABLE MEMORY"))/8;"(reals)"
80    OPTION BASE 1
90    DATA 1,2,4,7,11,16,22,29,37              ! Dielectric data for option
100   COM /Pass/ Relay                         ! for sharing to subs
110   COM /Pixel/ Chdr$[80],Dhdr$[80],INTEGER Lxtnt,Pixl(1:180,1:180)
120   DIM Hpiv(1:202),Hpr(1:20302)             ! dim to reasonable size
130   DIM Hdr$[80]                             ! available string for headers
140   DIM Admt(0:3),Admsav(0:3)                ! neighbor admittance values
150   INTEGER Xt(0:3),Yt(0:3)                  ! neighbor addresses
160   COM /Memr/ Graf(1:256,1:4),Ahdr$[80],Bhdr$[80],INTEGER Rep,Kwd !trials mem
170   !***> COM areas can be reaccessed with next RUN if identical name & sizes
180   !***> nb., max Lside >= .5 + sqrt(.25 + 2*(max dim - 1) )
190   LET Start=TIMEDATE
200   INTEGER Lside,Kond,Nodesz
210   INTEGER Ptrn,Nd,Nd1,Nd2,Xkin,Ykin,Xcnt,Ycnt,Xaddr,Yaddr,Boxes,Slant,Sprss
220   INTEGER Qdrnt,Rptr,Trans,Pose,Grpt,Tls,Sctr,Itmp,Occp
230   INTEGER Hzmax,Hxymax,Hpremax,Hopped,Hsteps,Hnd,Hcnt,Zcnt,Znd
240   INTEGER Cnmb,Qmat,Hnd1,Hnd2,Hleft,Hright
250   DIM Diel(1:9),Frpx(0:9),Msd$[60]
260   DIM Fln$[60]
270  !DIM Hpiv(1:128),Hpr(1:8192)  ! set to max physical storage
280   LET Grpt=1                            ! Initialize the data storage counter
290   ! * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - *
300   !    for which the integer variables roles are:
310   !     Relay = an available common pass variable
320   !     Lside = the # of pixel capacitor elements encounter along an edge
330   !               of the square of pixels
340   !     Tls = Lside or Lside/2 if 2x2 tiling
350   !     Qdrnt =  quadrant pixel array expanding switch, 0=off & 1=on
360   !     Px_tot = total # of pixels in square = Lside*Lside
370   !     Kond = the boundary condition on the sides of the overall composite
380   !             capacitor, 1) insulating sides or 2) periodic or sides which
390   !             wrap around
400   !     Nodesz = the maximum number of interaction nodes in forming
410   !               network, with 0 as the ground or base plate, 1 as center
420   !               node, and the final node number for the top plate.
430   !               Its value is: L*L/2-L+2 + (IF Periodic=1)*(L/2-1).
440   !
```

```
450   !                     OVERALL CAPACITOR FROM NODES
460   !                                !
470   !                                !
480   !                    ==================== top node or plate
490   !                     X X X X X X X X X X
500   !                      X X X X X X X X X
510   !                     X X X X X X X X X X
520   !                      X X X X X X X X X
530   !                     X X X X X X X X X X  center node at midpoint
540   !                      X X X X X X X X X
550   !                     X X X X X X X X X X
560   !                      X X X X X X X X X
570   !                     X X X X X X X X X X
580   !                    ==================== base node or plate
590   !                             __!__
600   !                              ___
610   !                               _              ...  X's represent nodes
620   !
630   !    Pixl( ) = the overlaying matrix representing the capacitor pixels
640   !    Dsplc( ) = displacement current of pixel per normalized volt/meter
650   !    Potnt( ) = pixel voltage relative to one volt across entire sample
660   !    Diel( ) = dielectric value or admittance value of a capacitor pixel
670   !             attached to addresses represented in the pixel grid
680   !    Frpx( ) = volume fractions associated with pixel types
690   !    Xt( ),Yt( ) = neighbor addresses
700   !    Fln$ = string refering to a filename, Hdr$ = 80 chars
710   !    Ahdr$,Bhdr$ = headers of 80 chrs for Data Title & ID for COM /Memr/
720   !    Chdr$,Dhdr$ = headers of 80 chrs for Pixel Title & ID for COM /Pixel/
730   !    Ptrn = choice of pixel grid filling pattern
740   !    Nd = a single number label for a node
750   !    Nd1,Nd2 = refers to a 1st node & a 2nd node @ specified by single
760   !              node numbers
770   !    Xkin,Ykin = kinship 2D address of a node number i.e. (x,y)
780   !    Xcnt,Ycnt = step counters to pixels neighbouring a node in 2D
790   !    Xaddr,Yaddr = addresses of neighbouring pixels in 2D
800   !    Boxes = total concentric boxes fitting within pixel grid or
810   !            number of 2x2 cell blocks along an edge on pixel grid
820   !    Slant = 0 if foward slash or 1 if backslash slanting capacitor
830   !    Rep,Rptr = overall number of repeats, Kwd=# of data storage types
840   !    Grpt = overall plus transpose repeats for use of data storage
850   !    Sprss = Suppression of printout details
860   !    Trans,Pose = Pixel transpose selection
870   !    Resp = Overall dielectric response of pixel sample along E
880   !    Tmp,Tmp1,Tmp2,Vt1,Vt2 = reals available for various uses
890   ! * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - * - *
900   PRINT
910   PRINT " > > > Happy capacitor composite adventures in 2 dimensions < < <"
920   PRINT "                  preformed on ";DATE$(TIMEDATE);
930   PRINT " at ";TIME$(TIMEDATE)
940   PRINT
950   ! *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
960   ! The hopper reduction subarray:                      S. Wallin, July 1990
970   !      ____  . . . . . . . . . .  > large symmetric sparse matrix .
980   !     | 1,1|                                                      .
```

```
990   !    |_#1_|____                                                        .
1000  !    | 2,1| 2,2|                                                       .
1010  !    |_#2_|_#3_|____                                                   .
1020  !    | 3,1| 3,2| 3,3|                                                  .
1030  !    |_#4_|_#5_|_#6_|                                                  .
1040  !    .                    ..        The hopper subarray moves          .
1050  !    .                         ..   down as pivoting progresses        .
1060  !   \/ lg sym sparse matrix         ..                                 .
1070  !    .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
1080  ! The "H" prefix is mainly used to denote variable use in hopper program
1090  ! Hx.. = row address of array element
1100  ! Hy.. = column address of array element
1110  !   i.e. (Hx..,Hy..) specifies a coordinate position
1120  ! Hz.. = the storage number associated with (Hx..,Hy..)
1130  ! Hxymax = altitude or base or diag # elements of reduction hopper
1140  ! Hzmax = the total # of elements contained within reduction hopper
1150  ! Hpremax = same as Hzmax but less Hxymax (ie less largest row)
1160  ! Hsteps = extent of larger interaction matrix upon hopper reduces
1170  ! Hleft,Hright = node #s of insulated sides Pixel grid
1180  ! Hpiv() = Pivoting vector of node reduction
1190  ! Hpr() = working hopper array of matrix reduction
1200  !  *    *    *    *    *    *    *    *    *    *    *    *    *    *    *    *    *    *
1210  PRINT ">Try hopping along to a solution of sparse matrices at ";
1220  DISP "IO to be: 0)default 1)lab 3.5"" 2)lab hardisc 3,4)A,B office ";
1230  INPUT "5)user defined",Nd
1240  IF Nd<0 THEN STOP
1250  IF Nd=0 THEN Msd$=""
1260  IF Nd=1 THEN Msd$=":CS80,700,1"
1270  IF Nd=2 THEN Msd$=":CS80,700"
1280  IF Nd=3 THEN Msd$=":CS80,703,0"
1290  IF Nd=4 THEN Msd$=":CS80,703,1"
1300  IF Nd=5 THEN INPUT "Name (completely) storage?",Msd$
1310  IF Msd$<>"" THEN PRINT RPT$(" ",50);"storage";Msd$
1320  PRINT " The pattern choices are:"
1330  PRINT "  0) internal, via COM /Memr/"
1340  PRINT "  1) from file storage"
1350  PRINT "  2) every pixel filled by user"
1360  PRINT "  3) random (i.e. well mixed)"
1370  PRINT "  4) by slanted fill level"
1380  PRINT "  5) with  an circle or ellipse of which can be tilted"
1390  PRINT "  6) with strata"
1400  PRINT "  7) concentric boxes"
1410  PRINT "  8) an ellipse with host & inclusion (2 components only,";
1420  PRINT " but symm wrt 1/2 vol)"
1430  INPUT "Select design of pixel grid? (see above)",Ptrn
1440  IF Ptrn<0 THEN STOP
1450  DISP " & boundary conditions? 1) Insulative ";
1460  INPUT "2) Wrap around or periodic",Kond
1470  IF Kond<0 THEN STOP
1480  IF Kond=0 THEN Kond=1
1490  IF Kond>2 THEN Kond=1+BIT(Kond+1,0)
1500  INPUT "Use 2x2 tiles on pixel grid? 0) No 1) Yes",Tls
1510  IF Tls<0 THEN STOP
1520  LET Tls=1+(Tls=1)
```

```
1530  INPUT "Quad fold symmetry expansion of pixel grid? 0) No 1) Yes",Qdrnt
1540  IF Qdrnt<0 THEN STOP
1550  LET Qdrnt=1+(Qdrnt=1)
1560  SELECT Ptrn
1570  CASE =0
1580    IF Lxtnt<2 THEN
1590      PRINT " Are Pixels there in memory? ..idled ..start again"
1600      STOP
1610    END IF
1620    IF Lxtnt>181 THEN PRINT " .. there may be too many Pixels"
1630    REDIM Pixl(1:Lxtnt,1:Lxtnt)
1640    LET Lside=Lxtnt*Qdrnt*Tls
1650    PRINT " From internal memory via COM, Pixels";Lxtnt;"x";Lxtnt;", title,"
1660    IF Chdr$<>"" THEN PRINT Chdr$
1670    IF Dhdr$<>"" THEN PRINT Dhdr$
1680  CASE =1                               ! Get pixels from file
1690    INPUT " Enclose (in """"s) file name to contain pixel pattern?",Fln$
1700    IF Fln$="" THEN STOP
1710    IF POS(Fln$,":")=0 THEN Fln$=Fln$&Msd$
1720    DISP " File named """;Fln$;""" ([";LEN(Fln$);"] characters)";
1730    DISP " is being read from storage"
1740    ASSIGN @Pixsrc TO Fln$;FORMAT OFF
1750    ENTER @Pixsrc;Chdr$,Dhdr$;Lxtnt     ! NB header assigned length of 80
1760    PRINT " Pixels contained in file """;Fln$;""", entitled with"
1770    PRINT Chdr$
1780    PRINT Dhdr$
1790    REDIM Pixl(1:Lxtnt,1:Lxtnt)         ! read initial Pixl(*) array
1800    ENTER @Pixsrc;Pixl(*)               ! retreive pixels from file
1810    ASSIGN @Pixsrc TO *                 ! close file
1820    LET Lside=Lxtnt*Qdrnt*Tls           ! actual Pixel side anticipated
1830    PRINT
1840  CASE ELSE                             ! Generate pixels
1850    DISP "How big a capacitor pixel grid in elements/side? ";
1860    INPUT "(even #, max ~180 int addr lmt)",Lside
1870    IF Lside<0 THEN STOP
1880    IF Lside<>SHIFT(SHIFT(Lside,1),-1) THEN
1890      PRINT " Odd";Lside;"Pixel length changed to even";
1900      LET Lside=SHIFT(SHIFT(Lside,1),-1)
1910      PRINT Lside
1920    END IF
1930    IF Lside=0 THEN Lside=2
1940    LET Lxtnt=Lside                      ! initial Pixel side length
1950    LET Lside=Lside*Qdrnt*Tls            ! Pixel side length anticipated
1960    IF Lside>182 THEN PRINT " ... near integer addressing limit"
1970  END SELECT                             ! end Ptrn test
1980  PRINT
1990  IF Tls=2 OR Qdrnt=2 THEN PRINT " Pixels now measures";Lside;"x";Lside
2000  PRINT " Pattern=";Ptrn;")";Lside;"x";Lside;
2010  IF Tls=1 THEN PRINT "pixels,";
2020  IF Tls=2 THEN PRINT "tiled pixels,";
2030  IF Kond=1 THEN PRINT " insulated or ""D"" field parallel to edge."
2040  IF Kond=2 THEN PRINT " periodic or voltage wrapping around at edges."
2050  LET Nodesz=SHIFT(Lside*Lside,1)-Lside+2+(Kond=2)*(SHIFT(Lside,1)-1)
2060 !ALLOCATE REAL Dsplc(Lside,Lside),Potnt(Lside,Lside) ! to be programmed
```

```
2070 !***> Initializing
2080 MAT Diel= (0)
2090 DISP "Dielectric sources? ";
2100 DISP "1) data input(programmed) 2) keyboard ";
2110 INPUT "3) by progression",Nd
2120 IF Nd=0 THEN Nd=2
2130 SELECT Nd
2140 CASE =1
2150   FOR Nd1=1 TO 9
2160     READ Diel(Nd1)
2170   NEXT Nd1
2180 CASE =2
2190   FOR Nd1=1 TO 9
2200     DISP "Give dielectric value at";Nd1;
2210     INPUT "? (or enter negative if to cease)",Diel(Nd1)
2220     IF Diel(Nd1)<0 THEN
2230       LET Diel(Nd1)=0
2240       LET Nd1=9
2250     ELSE
2260       PRINT "diel[";Nd1;"]=";PROUND(Diel(Nd1),-4);",";
2270     END IF
2280   NEXT Nd1
2290 CASE =3
2300   INPUT "Dielectric value of pixel type ""[1]""?",Tmp
2310   DISP "Multiplier of progression for each succeeding value ";
2320   INPUT "to fill [2],[3],..,[9] ?",Tmp1
2330   FOR Nd1=1 TO 9
2340     Diel(Nd1)=Tmp
2350     Tmp=Tmp*Tmp1
2360     PRINT "diel[";Nd1;"]=";PROUND(Diel(Nd1),-4);",";
2370   NEXT Nd1
2380 CASE ELSE
2390   STOP
2400 END SELECT
2410 PRINT
2420 LET Sprss=1
2430 IF Nodesz<32 THEN
2440   INPUT "Surpress screen listing details, 0) No 1) Yes?",Sprss
2450   IF Sprss<0 THEN STOP
2460 END IF
2470 INPUT "Any overall repeats?",Rep
2480 IF Rep<0 THEN STOP
2490 IF Rep=0 THEN LET Rep=1
2500 INPUT "Desire transpose of pixel grid? 0) No 1) Yes",Pose
2510 IF Pose<0 THEN STOP
2520 LET Pose=1+BIT(Pose,0)
2530 REM " Solution acheived by a sparse matrix reduced pivoting technique"
2540 !***>> Overall repetition, may require additional editing
2550 LET Kwd=4             ! user has selected to program for 4 data columns
2560 REDIM Graf(1:Rep,1:Kwd)
2570 FOR Rptr=1 TO Rep
2580   LET Relay=Rptr  !7/27/90
2590   MAT Frpx= (0)
2600   !**> if Ptrn=0 internal or Ptrn=1 then Pixels  read from file
```

```
2610     IF Ptrn=2 THEN CALL Pixl2d_fill
2620     IF Ptrn=3 THEN CALL Pixl2d_rand
2630     IF Ptrn=4 THEN CALL Pixl2d_tilt
2640     IF Ptrn=5 THEN CALL Pixl2d_ellps
2650     IF Ptrn=6 THEN CALL Pixl2d_strat
2660     IF Ptrn=7 THEN CALL Pixl2d_cbox
2670     IF Ptrn=8 THEN CALL Pixl2d_ellp2
2680     IF Tls=2 THEN
2690       LET Xkin=SIZE(Pixl,1)            ! redimensioning
2700       LET Ykin=SIZE(Pixl,2)
2710       LET Nd1=Xkin+Ykin
2720       LET Nd2=SHIFT(Xkin*Ykin,-2)      ! 4*Xkin*Ykin
2730       REDIM Pixl(1:1,1:Nd2)
2740       FOR Xcnt=(Xkin-1) TO 0 STEP -1
2750         FOR Ycnt=Ykin TO 1 STEP -1
2760           LET Pixl(1,Xcnt*2*Ykin+Ycnt)=Pixl(1,Xcnt*Ykin+Ycnt)
2770         NEXT Ycnt
2780       NEXT Xcnt
2790       REDIM Pixl(1:Nd1,1:Nd1)          ! set new array dimen
2800       FOR Xcnt=Xkin TO 1 STEP -1       ! tiling 2x2
2810         FOR Ycnt=Ykin TO 1 STEP -1
2820           LET Itmp=Pixl(Xcnt,Ycnt)
2830           LET Xaddr=SHIFT(Xcnt,-1)     ! effective 2* op
2840           LET Yaddr=SHIFT(Ycnt,-1)
2850           LET Pixl(Xaddr,Yaddr)=Itmp
2860           LET Pixl(Xaddr-1,Yaddr)=Itmp
2870           LET Pixl(Xaddr,Yaddr-1)=Itmp
2880           LET Pixl(Xaddr-1,Yaddr-1)=Itmp
2890         NEXT Ycnt
2900       NEXT Xcnt
2910     END IF
2920     IF Qdrnt=2 THEN
2930       LET Nd2=SIZE(Pixl,1)+SIZE(Pixl,2)
2940       LET Nd1=SHIFT(Nd2,1)
2950       REDIM Pixl(1:Nd2,1:Nd2)          ! redim to dble quad duplic
2960       FOR Xcnt=Nd1 TO 1 STEP -1
2970         LET Xaddr=Nd2+1-Xcnt           ! quad complement X counter
2980         LET Xkin=SHIFT(Xcnt+1,1)       ! effectively DIV 2 op
2990         LET Ykin=BIT(Xcnt+1,0)         ! effectively odd<=>even op
3000         FOR Ycnt=1 TO Nd1
3010           LET Yaddr=Nd2+1-Ycnt         ! quad complement Y counter
3020           LET Itmp=Pixl(Xkin,Ycnt+Nd1*Ykin)
3030           LET Pixl(Xcnt,Ycnt)=Itmp     ! Itmp takes care of redim elements
3040           LET Pixl(Xaddr,Ycnt)=Itmp
3050           LET Pixl(Xcnt,Yaddr)=Itmp
3060           LET Pixl(Xaddr,Yaddr)=Itmp
3070         NEXT Ycnt
3080       NEXT Xcnt
3090     END IF
3100     LET Lside=SIZE(Pixl,1)            ! update Pixl extent along edge
3110     LET Px_tot=Lside*Lside
3120     LET Boxes=SHIFT(Lside,1)
3130     IF Boxes<1 THEN PRINT "WARNING! may be too small of a pixel grid"
3140     LET Nodesz=SHIFT(Lside*Lside,1)-Lside+2+(Kond=2)*(SHIFT(Lside,1)-1)
```

```
3150    !***> Tranpose of Pixel grid
3160    LET Trans=Pose ! if loop then use next line
3170   !FOR Trans=1 TO Pose
3180    IF Trans=2 THEN                      ! Tranpose
3190      PRINT " TRANSPOSING"
3200      FOR Xcnt=1 TO Lside
3210        FOR Ycnt=(Xcnt+1) TO Lside
3220          LET Itmp=Pixl(Ycnt,Xcnt)    ! swap Xcoordinate<->Ycoordinate
3230          LET Pixl(Ycnt,Xcnt)=Pixl(Xcnt,Ycnt)
3240          LET Pixl(Xcnt,Ycnt)=Itmp
3250        NEXT Ycnt
3260      NEXT Xcnt
3270    END IF
3280    !***> Evaluation of pixel type volume fractions
3290    IF NOT (Sprss) OR Lside<81 THEN PRINT "  Pixels";Lside;"x";Lside
3300    FOR Xcnt=Lside TO 1 STEP -1
3310      FOR Ycnt=1 TO Lside
3320        Frpx(Pixl(Xcnt,Ycnt))=Frpx(Pixl(Xcnt,Ycnt))+1
3330        IF NOT (Sprss) OR Lside<40 THEN PRINT " ";VAL$(Pixl(Xcnt,Ycnt));
3340      NEXT Ycnt
3350      IF NOT (Sprss) OR Lside<40 THEN PRINT
3360    NEXT Xcnt
3370    MAT Frpx= Frpx/(Px_tot)
3380    PRINT " Volume %s: ";
3390    FOR Nd=1 TO 9
3400      IF Frpx(Nd)<>0 THEN PRINT PROUND(100*Frpx(Nd),-1);"%=>[";Nd;"],";
3410    NEXT Nd
3420    PRINT
3430    IF Frpx(0)<>0 THEN PRINT "WARNING! check pixels"
3440    DISP " .. wait";Rptr;"of";Rep;".. solving node interact matrix,";
3450    DISP Nodesz;"by";Nodesz;"from time ";TIME$(TIMEDATE)
3460    LET Tmp=TIMEDATE                     ! Benchmarker
3470   !***>  HOP technique of sparse matrix reduction
3480    PRINT " Solving INTERACTION matrix";Nodesz;"x";Nodesz;"via hopper"
3490    LET Hxymax=Lside                     ! size of Hopper for matrix reduction
3500    LET Hzmax=(1.0+Hxymax)*Hxymax DIV 2     ! memory reduction capabilities
3510    LET Hsteps=Nodesz
3520    LET Hpremax=Hzmax-Hxymax
3530    REDIM Hpiv(1:Hxymax),Hpr(1:Hzmax)
3540    LET Hleft=1                         ! markers of node #s on left insl BC
3550    LET Hright=Boxes                    ! markers of node #s on right insl BC
3560    MAT Hpiv= (0)
3570    MAT Hpr= (0)
3580    LET Znd=0                           ! to previous row tri-diag accum
3590    FOR Hnd=1 TO Hxymax                 ! Filling hopper work array
3600      IF NOT (Sprss) THEN PRINT " node's ";VAL$(Hnd);" neighbors are";
3610      FOR Sctr=0 TO 3                   ! Diagonal or self interact terms
3620        IF Kond=1 THEN CALL Cvndi(Hnd,Lside,Sctr,Xt(Sctr),Yt(Sctr))
3630        IF Kond=2 THEN CALL Cvndp(Hnd,Lside,Sctr,Xt(Sctr),Yt(Sctr))
3640        LET Admt(Sctr)=Diel(Pixl(Xt(Sctr),Yt(Sctr)))
3650        IF Kond=1 THEN                  ! adj for insl BC on Pixel grid
3660          IF Hnd=Hleft THEN             ! test if left side node
3670            IF NOT (BIT(Sctr,0)) THEN
3680              IF BIT(Sctr,1) THEN   ! make series admit combo
```

```
3690                IF Xt(2)<Lside THEN
3700                  LET Admsav(0)=Diel(Pixl(Xt(2)+1,1))
3710                  LET Admt(2)=Admt(2)*Admsav(0)/(Admt(2)+Admsav(0))
3720                  LET Admsav(2)=Admt(2) ! save for later use
3730                END IF
3740              ELSE                      ! reuse former upper admit combo
3750                IF Hnd>1 THEN LET Admt(0)=Admsav(2) ! pass by on 1st node
3760              END IF                    ! end of Sctr=0,2 test
3770            END IF                      ! end of insl BC Pixel test
3780          END IF                        ! end of Hnd=Hleft test
3790          IF Hnd=Hright THEN            ! test if right side node
3800            IF BIT(Sctr,0) THEN
3810              IF BIT(Sctr,1)=1 THEN ! make series admit combo
3820                IF Xt(3)<Lside THEN
3830                  LET Admsav(1)=Diel(Pixl(Xt(3)+1,Lside)) ! cmbn above ngbr
3840                  LET Admt(3)=Admt(3)*Admsav(1)/(Admt(3)+Admsav(1))
3850                  LET Admsav(3)=Admt(3) ! save for later use
3860                END IF
3870              ELSE                      ! reuse former admit combo
3880                IF Hnd>Boxes THEN LET Admt(1)=Admsav(3) ! Skip 1st pass
3890              END IF                    ! end of Sctr=0 test
3900            END IF                      ! end of insl BC Pixel test
3910          END IF                        ! end of Hnd=Hright test
3920        END IF                          ! end if for Kond=1  test
3930        LET Hpr(Znd+Hnd)=Hpr(Znd+Hnd)+Admt(Sctr)
3940        IF NOT (Sprss) THEN PRINT " (";VAL$(Xt(Sctr));",";VAL$(Yt(Sctr));")"
3950      NEXT Sctr
3960      SELECT Kond
3970      CASE =1
3980        LET Hnd1=FNNi(Xt(0),Yt(0),Lside,1)
3990        LET Hnd2=FNNi(Xt(1),Yt(1),Lside,1)
4000      CASE =2
4010        LET Hnd1=FNNp(Xt(0),Yt(0),Lside,1)
4020        LET Hnd2=FNNp(Xt(1),Yt(1),Lside,1)
4030      CASE ELSE
4040        PRINT " out of bounds, boundary condition, in HOPper"
4050      END SELECT
4060      IF NOT (Sprss) THEN PRINT " w/ lower nodes";Hnd1;"&";Hnd2
4070      IF Hnd1<0 OR Hnd2<0 THEN PRINT " Warning node <=> addresses?"
4080      IF Hnd1<Hnd AND Hnd1>0 THEN
4090        LET Hpr(Znd+Hnd1)=Hpr(Znd+Hnd1)-Admt(0)
4100      END IF
4110      IF Hnd2<Hnd AND Hnd2>0 THEN
4120        LET Hpr(Znd+Hnd2)=Hpr(Znd+Hnd2)-Admt(1)
4130      END IF
4140      IF Kond=1 THEN                    ! increment insl BC left & right node #
4150        IF Hnd=Hleft THEN LET Hleft=Hleft+Lside-1
4160        IF Hnd=Hright THEN LET Hright=Hright+Lside-1
4170      END IF
4180      LET Znd=Znd+Hnd                   ! loop count accumulator
4190    NEXT Hnd                            ! end of set up of work matrix
4200    LET Zcnt=0
4210    IF NOT (Sprss) THEN
4220      FOR Xcnt=1 TO Hxymax              ! printout HOPper
```

```
4230         FOR Ycnt=1 TO Xcnt
4240           PRINT PROUND(Hpr(Zcnt+Ycnt),-3),
4250         NEXT Ycnt
4260         PRINT
4270         LET Zcnt=Zcnt+Xcnt
4280       NEXT Xcnt
4290     END IF
4300     FOR Hopped=1 TO Hsteps-1           ! Let the pivoting begin, & drip dry
4310       LET Hnd=Hxymax+Hopped            ! count of oncoming node number
4320       LET Hpiv(1)=1                    ! normalize to 1st elment pivot vectr
4330       LET Hnrm=1/Hpr(1)                ! normalizing multiplier
4340       LET Zcnt=2                       ! convert array storage for 1st colmn
4350       FOR Hcnt=2 TO Hxymax             ! set pivot vector
4360         LET Hpiv(Hcnt)=Hpr(Zcnt)*Hnrm
4370         LET Zcnt=Zcnt+Hcnt
4380       NEXT Hcnt
4390       !***> one can output pivot here for backsub later
4400       IF NOT (Sprss) THEN
4410         PRINT " At reduction";Hopped;"the pivots are:"
4420         FOR Hcnt=Hxymax TO 1 STEP -1
4430           PRINT PROUND(Hpiv(Hcnt),-4);
4440         NEXT Hcnt
4450         PRINT
4460       END IF
4470       LET Zcnt=1                       ! initialize filling counter
4480       FOR Xcnt=2 TO Hxymax             ! heart of pivoting
4490         IF Hpiv(Xcnt)<>0 THEN          ! sparseness efficiency =0, no-op
4500           FOR Ycnt=2 TO Xcnt           ! adj each array row with pivot vectr
4510             IF Hpiv(Ycnt)<>0 THEN      ! sparseness efficiency =0, no-op
4520               LET Hpr(Zcnt+Ycnt)=Hpr(Zcnt+Ycnt)-Hpr(Zcnt+1)*Hpiv(Ycnt)
4530             END IF
4540           NEXT Ycnt
4550         END IF
4560         LET Zcnt=Zcnt+Xcnt
4570       NEXT Xcnt
4580       LET Zcnt=0                       ! initialize filling counter lower
4590       FOR Xcnt=1 TO Hxymax-1
4600         FOR Ycnt=1 TO Xcnt             ! hopping along for shake up
4610           LET Hpr(Zcnt+Ycnt)=Hpr(Zcnt+Ycnt+1+Xcnt)
4620         NEXT Ycnt
4630         LET Zcnt=Zcnt+Xcnt
4640       NEXT Xcnt
4650       FOR Ycnt=1 TO Hxymax             ! feed hopper, clear last row
4660         LET Hpr(Hpremax+Ycnt)=0
4670       NEXT Ycnt
4680       SELECT Hnd
4690       CASE <Hsteps                     ! feed unless over lg array extent
4700         IF NOT (Sprss) THEN PRINT " node's ";VAL$(Hnd);" neighbors are";
4710         FOR Sctr=0 TO 3                ! with diagonal or self interact terms
4720           IF Kond=1 THEN CALL Cvndi(Hnd,Lside,Sctr,Xt(Sctr),Yt(Sctr))
4730           IF Kond=2 THEN CALL Cvndp(Hnd,Lside,Sctr,Xt(Sctr),Yt(Sctr))
4740           LET Admt(Sctr)=Diel(Pixl(Xt(Sctr),Yt(Sctr)))
4750           IF Kond=1 THEN               ! adj for insl BC on Pixel grid
4760             IF Hnd=Hleft THEN          ! test if left side node
```

```
4770               IF NOT (BIT(Sctr,0)) THEN
4780                 IF BIT(Sctr,1) THEN ! make series admit combo
4790                   IF Xt(2)<Lside THEN
4800                     LET Admsav(0)=Diel(Pixl(Xt(2)+1,1))
4810                     LET Admt(2)=Admt(2)*Admsav(0)/(Admt(2)+Admsav(0))
4820                     LET Admsav(2)=Admt(2)! save for later use
4830                   END IF
4840                 ELSE                  ! reuse former upper admit combo
4850                   IF Hnd>1 THEN LET Admt(0)=Admsav(2)! pass by on 1st node
4860                 END IF                ! end of Sctr=0,2 test
4870               END IF                  ! end of insl BC Pixel test
4880             END IF                    ! end of Hnd=Hleft test
4890             IF Hnd=Hright THEN        ! test if right side node
4900               IF BIT(Sctr,0) THEN
4910                 IF BIT(Sctr,1)=1 THEN ! make series admit combo
4920                   IF Xt(3)<Lside THEN
4930                     LET Admsav(1)=Diel(Pixl(Xt(3)+1,Lside))! cmbn above ngbr
4940                     LET Admt(3)=Admt(3)*Admsav(1)/(Admt(3)+Admsav(1))
4950                     LET Admsav(3)=Admt(3)! save for later use
4960                   END IF
4970                 ELSE                  ! reuse former admit combo
4980                   IF Hnd>Boxes THEN LET Admt(1)=Admsav(3)! Skip 1st pass
4990                 END IF                ! end of Sctr=0 test
5000               END IF                  ! end of insl BC Pixel test
5010             END IF                    ! end of Hnd=Hright test
5020           END IF                      ! end if for Kond=1  test
5030           LET Hpr(Hzmax)=Hpr(Hzmax)+Admt(Sctr)
5040           IF NOT (Sprss) THEN PRINT " (";VAL$(Xt(Sctr));",";VAL$(Yt(Sctr));"
5050         NEXT Sctr
5060         SELECT Kond                     ! with off diagonal terms
5070         CASE =1
5080           LET Hnd1=FNNi(Xt(0),Yt(0),Lside,1)
5090           LET Hnd2=FNNi(Xt(1),Yt(1),Lside,1)
5100         CASE =2
5110           LET Hnd1=FNNp(Xt(0),Yt(0),Lside,1)
5120           LET Hnd2=FNNp(Xt(1),Yt(1),Lside,1)
5130         END SELECT                      ! to SELECT Kond
5140         IF NOT (Sprss) THEN PRINT " w/ lower nodes";Hnd1;"&";Hnd2
5150         IF Hnd1<0 OR Hnd2<0 THEN PRINT " Warning node <=> addresses?"
5160         IF Hnd1<Hnd AND Hnd1>0 THEN
5170           LET Hnd1=Hnd1+Hzmax-Hnd
5180           LET Hpr(Hnd1)=Hpr(Hnd1)-Admt(0)
5190         END IF
5200         IF Hnd2<Hnd AND Hnd2>0 THEN
5210           LET Hnd2=Hnd2+Hzmax-Hnd
5220           LET Hpr(Hnd2)=Hpr(Hnd2)-Admt(1)
5230         END IF
5240       CASE =Hsteps                      ! top node is contact node
5250         FOR Ycnt=1 TO Lside             ! diagonal
5260           Hpr(Hzmax)=Hpr(Hzmax)+Diel(Pixl(Lside,Ycnt))
5270           LET Hnd1=Hzmax-Boxes+SHIFT(Ycnt-1,1)   ! off diagonal node #
5280           LET Hpr(Hnd1)=Hpr(Hnd1)-Diel(Pixl(Lside,Ycnt))
5290         NEXT Ycnt
5300       END SELECT                        ! to SELECT Hnd
```

```
5310      IF NOT (Sprss OR Hnd>Hsteps) THEN
5320        FOR Ycnt=1 TO Hxymax            ! cont printout of HOPper
5330          PRINT PROUND(Hpr(Hpremax+Ycnt),-3),
5340        NEXT Ycnt
5350        PRINT
5360      END IF
5370      IF Kond=1 THEN                    ! increment insl BC left & right node #
5380        IF Hnd=Hleft THEN LET Hleft=Hleft+Lside-1
5390        IF Hnd=Hright THEN LET Hright=Hright+Lside-1
5400      END IF
5410    NEXT Hopped
5420    IF NOT (Sprss) THEN PRINT " Hopper funnels down to";Hpr(1)
5430    !***> Hpr(1) contains the end of the interaction reduction
5440    !   LET Hpiv(Hsteps)=1/Hpr(1)       ! backsubstitute for solution vector
5450    !   FOR Hcnt=(Hsteps-1) TO 1 STEP -1
5460    !     LET Hpiv(Hcnt)=0
5470    !     FOR Zcnt=Hcnt TO Hsteps
5480    !      LET Hpiv(Hcnt)=Hpiv(Hcnt)-Pivotstorg(Hcnt,1+Zcnt-Hcnt)*Hpiv(Zcnt)
5490    !     NEXT Zcnt
5500    !   NEXT Hcnt
5510    PRINT "                ... at ";TIME$(TIMEDATE);" inversion excution ";
5520    PRINT "time took";PROUND(TIMEDATE-Tmp,-1);"seconds"
5530    DISP
5540    LET Resp=Hpr(1)                     ! principal diel resp
5550   !***> Pixl displacement field/current & potentials
5560    !  FOR Ycnt=1 TO Lside
5570    !    FOR Xcnt=1 TO Lside
5580    !      Slant=(Xcnt+Ycnt) MOD 2
5590    !      Xaddr=Xcnt-Boxes-Slant        ! (x,y) of node upper to pixel
5600    !      Yaddr=Ycnt-Boxes
5610    !      CALL Xy_to_node(Nd1,Xaddr,Yaddr,Lside,Kond)
5620    !      Xaddr=Xcnt-Boxes+Slant-1      ! (x,y) of node lower to pixel
5630    !      Yaddr=Ycnt-Boxes-1
5640    !      CALL Xy_to_node(Nd2,Xaddr,Yaddr,Lside,Kond)
5650    !      IF Nd1<>Nd2 AND Nd1>0 AND Nd2>0 THEN
5660    !        Dsplc(Xcnt,Ycnt)=Diel(Pixl(Xcnt,Ycnt))*(Hpiv(Nd1)-Hpiv(Nd2))
5670    !        Potnt(Xcnt,Ycnt)=(Hpiv(Nd1)+Hpiv(Nd2))/2
5680    !      END IF
5690    !      IF Nd2=0 AND Nd1>0 THEN
5700    !        Dsplc(Xcnt,Ycnt)=Diel(Pixl(Xcnt,Ycnt))*Hpiv(Nd1)
5710    !        Potnt(Xcnt,Ycnt)=Hpiv(Nd1)/2
5720    !      END IF
5730    !    NEXT Xcnt
5740    !  NEXT Ycnt
5750    !***> additional modification of Potential & Displacement array fields
5760    !  IF Kond=1 THEN
5770    !    FOR Ycnt=2 TO (Lside-2) STEP 2
5780    !      FOR Nd=-1 TO 1 STEP 2
5790    !        LET Xcnt=(Nd+1)*Boxes+(Nd=-1)  ! (Xcnt,Ycnt) refer to pixel
5800    !        LET Xaddr=Nd*(Boxes-1)          ! (Xaddr,Yaddr) refer to node
5810    !        LET Yaddr=Ycnt-Boxes
5820    !        CALL Xy_to_node(Nd1,Xaddr,Yaddr+1,Lside,Kond)! node # upper
5830    !        CALL Xy_to_node(Nd2,Xaddr,Yaddr-1,Lside,Kond)! node # lower
5840    !        IF Nd1<>Nd2 AND Nd1>0 AND Nd2>0 THEN ! Evaluate along side nodes
```

```
5850  !            LET Tmp1=Diel(Pixl(Xcnt,Ycnt+1))! Upper dielectric pixel
5860  !            LET Tmp2=Diel(Pixl(Xcnt,Ycnt))! Lower dielectric pixel
5870  !            LET Vt1=Hpiv(Nd1)
5880  !            LET Vt2=Hpiv(Nd2)
5890  !            IF Tmp1<>0 AND Tmp2<>0 THEN Tmp=(Vt1*Tmp1+Vt2*Tmp2)/(Tmp1+Tmp2
5900  !            LET Potnt(Xcnt,Ycnt+1)=(Vt1+Tmp)/2! Pixl volts
5910  !            LET Potnt(Xcnt,Ycnt)=(Vt2+Tmp)/2
5920  !            IF Tmp1<>0 AND Tmp2<>0 THEN
5930  !              LET Dsplc(Xcnt,Ycnt+1)=(Vt1-Vt2)/(1/Tmp1+1/Tmp2)
5940  !            END IF
5950  !            LET Dsplc(Xcnt,Ycnt)=Dsplc(Xcnt,Ycnt+1)! Displacement mag.
5960  !          END IF
5970  !        NEXT Nd
5980  !      NEXT Ycnt
5990  !    END IF
6000  !    MAT Potnt= Potnt*(Resp)      ! Normalizing to 1 volt across sample
6010  !    MAT Dsplc= Dsplc*(Resp)      ! & sum of displacements along row=diel
6020  !    LET Nd1=1                    ! sign provider for following loop
6030  !    FOR Xcnt=1 TO Lside
6040  !      LET Resp2=Resp2+Nd1*(Dsplc(Xcnt,Boxes)-Dsplc(Xcnt,Boxes+1))
6050  !      LET Nd1=-Nd1
6060  !    NEXT Xcnt
6070  !    LET Resp2=Resp2/2            ! dielectric response perp to E
6080  !***> Should be end of calculations, printouts follow
6090  !***> Printout of the dielectric pixel array
6100    IF NOT (Sprss) AND Lside<13 THEN
6110      PRINT "DIELECTRIC PIXEL ARRAY, 2-dimensional,";Lside;"by";Lside
6120      FOR Xcnt=Lside TO 1 STEP -1
6130        FOR Ycnt=1 TO Lside
6140          PRINT USING "DDDD.D,#";PROUND(Diel(Pixl(Xcnt,Ycnt)),-1)
6150        NEXT Ycnt
6160        PRINT
6170      NEXT Xcnt
6180    END IF
6190  !***> Printout of the hopper array
6200    !****>  Find the series<->parallel factor
6210    LET Tmp1=FNWnr(Diel(*),Frpx(*),Resp,Tmp,9)
6220    PRINT
6230    PRINT "Composite Dielectric Response Tensor Components:"
6240    PRINT " principal=";Resp
6250    PRINT "  & series<->parallel factor =";
6260    PRINT PROUND(Tmp1,-3);"(+/-";PROUND(Tmp,-4);"% iteration error)"
6270    PRINT
6280  !  IF Sprss=0 THEN
6290  !    PRINT "PIXEL VOLTAGES 2-dimensional,";Lside;"by";Lside
6300  !    Tmp=FNMatprnt(Potnt(*),-Lside)
6310  !    PRINT "PIXEL DISPLACEMENT FIELD MAGNITIUDES,";Lside;"by";Lside
6320  !    Tmp=FNMatprnt(Dsplc(*),-Lside)
6330  !  END IF
6340     !***> NOTE: Tranpose used then it is an additional cycle to Rptr
6350    Graf(Grpt,1)=Rptr+Tmp/100
6360    Graf(Grpt,2)=Frpx(1)
6370    Graf(Grpt,3)=Resp
6380    Graf(Grpt,4)=Tmp1
```

```
6390    LET Grpt=Grpt+1                    ! Increment storage counter
6400    !NEXT Trans
6410   NEXT Rptr
6420   !***> output repeat calculations
6430   LET Bhdr$="("&VAL$(Lside)&"x"&VAL$(Lside)&")"
6440   IF Tls=1 THEN LET Bhdr$=Bhdr$&" elmnts"
6450   IF Tls=2 THEN LET Bhdr$=Bhdr$&"/(2x2s)"
6460   IF Kond=1 THEN LET Bhdr$=Bhdr$&" InslBC"
6470   IF Kond=2 THEN LET Bhdr$=Bhdr$&" PrdcBC"
6480   LET Bhdr$=Bhdr$&" Sparse"     ! solution by sparse methods
6490   IF Qdrnt=2 THEN LET Bhdr$=Bhdr$&" 4fold"
6500   IF Ptrn=0 THEN Bhdr$=Bhdr$&" intrnl,"
6510   IF Ptrn=1 THEN Bhdr$=Bhdr$&" "&Fln$
6520   IF Ptrn=2 THEN Bhdr$=Bhdr$&" USER,"
6530   IF Ptrn=3 THEN Bhdr$=Bhdr$&" RANDOM,"
6540   IF Ptrn=4 THEN Bhdr$=Bhdr$&" SLANT,"
6550   IF Ptrn=5 THEN Bhdr$=Bhdr$&" ELLIPSE,"
6560   IF Ptrn=6 THEN Bhdr$=Bhdr$&" STRAT,"
6570   IF Ptrn=7 THEN Bhdr$=Bhdr$&" BOXES,"
6580   LET Occp=LEN(Bhdr$)
6590   LET Bhdr$[1+Occp]=RPT$(" ",80-Occp)   ! pad with blanks
6600   LET Bhdr$[60]=" "&DATE$(TIMEDATE)&","&TIME$(TIMEDATE)
6610   LET Dhdr$=Bhdr$
6620   IF Rep=1 THEN PRINT "  for the case abbreviated .."
6630   IF Rep=1 THEN PRINT Bhdr$
6640   IF Rep>1 THEN
6650     PRINT " Summary of";Rep;"repeat variations: (as programmed)"
6660     FOR Rptr=1 TO Rep
6670       PRINT " Case";((Rptr-1) DIV Pose)+1;")",PROUND(Graf(Rptr,1),-3),
6680       PRINT PROUND(Graf(Rptr,2),-3),PROUND(Graf(Rptr,3),-3),
6690       PRINT PROUND(Graf(Rptr,4),-3)
6700     NEXT Rptr
6710     DISP " Save repeat info (array form,";SIZE(Graf,1);"x";SIZE(Graf,2);
6720     INPUT ")? 0) No 1) Definitely",Nd1
6730     IF Nd1=1 THEN
6740       DISP " Enclose (in """"s) new file name to send info vectors to?";
6750       INPUT " (null=use old file)",Fln$
6760       IF POS(Fln$,":")=0 THEN Fln$=Fln$&Msd$
6770       INPUT " Title, (up to 80 characters)",Ahdr$
6780       LET Ahdr$[1+LEN(Ahdr$)]=RPT$(" ",80-LEN(Ahdr$))! pad with blanks
6790       DISP " File named """;Fln$;""" ([";LEN(Fln$);"] characters)";
6800       DISP " to contain repeat info"
6810       PRINT " File """;Fln$;"""'s user and description headers are ";
6820       PRINT "(2 lines):"
6830       PRINT Ahdr$
6840       PRINT Bhdr$
6850       IF Fln$="" THEN
6860         INPUT "  Enter the filename to be created? null=stop",Fln$
6870         IF Fln$="" THEN STOP
6880         DISP " Enter file""";Fln$;"""'s storage size limit in bytes (";
6890         DISP VAL$(256+8*Rep*Pose*Kwd);")";
6900         INPUT "?",Nd1
6910         IF Nd1<1048 THEN Nd1=1048     ! 1 kiloBYTE min
6920         CREATE Fln$,Nd1
```

```
6930      ELSE
6940        IF POS(Fln$,":")=0 THEN Fln$=Fln$&Msd$
6950      END IF
6960      ASSIGN @Infostr TO Fln$;FORMAT OFF
6970      OUTPUT @Infostr;Ahdr$,Bhdr$,Rep,Kwd,Graf(*),END
6980      ASSIGN @Nodstr TO *
6990    END IF
7000  END IF
7010  !***> Pixel file output choice
7020  LET Ndl=0
7030  IF Ptrn<>1 THEN INPUT " Save last pixel grid? 0)No 1)Yes",Ndl
7040  IF Ndl=1 THEN
7050    DISP " Enclose (in """"s) new file name to send pixel pattern to?";
7060    INPUT " (null=use old file)",Fln$
7070    IF POS(Fln$,":")=0 THEN Fln$=Fln$&Msd$
7080    INPUT " Title (up to 80 characters) if null then default label",Chdr$
7090    LET Ohdr$=Bhdr$
7100    DISP " File named """;Fln$;""" ([";LEN(Fln$);"] characters)";
7110    DISP " contains the pixel grid"
7120    PRINT " File """;Fln$;"""'s header is "
7130    PRINT Hdr$
7140    IF Fln$<>"" THEN
7150      DISP " Give file""";Fln$;"""'s max capacity limit in bytes";
7160      DISP "? (~";VAL$(128+SHIFT(Px_tot,-1));")";
7170      INPUT " ",Ndl
7180      IF Ndl<256 THEN Ndl=256
7190      CREATE Fln$,Ndl
7200    ELSE
7210      INPUT "  Enter the existing filename?",Fln$
7220      IF POS(Fln$,":")=0 THEN Fln$=Fln$&Msd$
7230    END IF
7240    ASSIGN @Pixstr TO Fln$;FORMAT OFF
7250    OUTPUT @Pixstr;Hdr$,Lside,Pixl(*),END
7260    ASSIGN @Pixstr TO *
7270  END IF
7280  !***> Interaction file output choice
7290  PRINT RPT$(" ",25);"...elapsed";PROUND(TIMEDATE-Start,-1);
7300  PRINT "sec for completion at ";TIME$(TIMEDATE)
7310  PRINT " MEMORY IS";VAL(SYSTEM$("AVAILABLE MEMORY"))/8;"(reals)"
7320  LET Lxtnt=Lside                        ! update COM /Pixel/ ie Pixl(*) size
7330  END
7340  ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
7350  DEF FNNi(INTEGER Xn,Yn,Lszn,Lup)
7360  ! Returns the node number for a square Pixel grid network
7370  ! of capacitors for case of insulated sides.
7380  ! (Xn,Yn) 2D coordinates of Pixel leading to the nearest node
7390  ! If Up=1 then Pixl above node, else Pixl below node
7400    INTEGER Xhf,Ysw,Ndb,Lyr
7410    LET Lup=BIT(Lup,0)
7420    LET Lyr=SHIFT(Lszn,1)            ! in essence divides by 2
7430    IF Xn>0 AND Xn<=Lszn AND Yn>0 AND Yn<=Lszn THEN
7440      LET Ndb=1
7450      IF Xn=1 AND Lup THEN LET Ndb=0
7460      IF Xn=Lszn AND NOT (Lup) THEN LET Ndb=SHIFT(Lszn*Lszn,1)-Lszn+2
```

```
7470      IF Ndb=1 THEN                     ! executes if node not yet assigned
7480        LET Xhf=SHIFT(Xn-1-Lup,1)       ! effectively DIV 2
7490        LET Xsw=(BIT(Xn,0) EXOR Lup)! 0=even 1=odd in Xn
7500        LET Ndb=Xhf*(Lszn-1)+SHIFT(Yn+Xsw,1)! basic node count
7510        IF NOT (Xsw) THEN LET Ndb=Ndb+Lyr
7520        IF (Yn=1) AND NOT (Xsw) THEN LET Ndb=Ndb+Lyr-Lup*(Lszn-1)
7530        IF Yn=Lszn AND NOT (Xsw) THEN LET Ndb=Ndb+Lyr-1-Lup*(Lszn-1)
7540      END IF
7550    ELSE
7560      LET Ndb=-1
7570    END IF
7580    RETURN Ndb
7590  FNEND
7600  ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
7610  DEF FNNp(INTEGER Xn,Yn,Lszn,Lup)
7620  ! Returns the node number for a square Pixel grid network
7630  ! of capacitors for case of periodic or wrap around sides
7640  ! (Xn,Yn) 2D coordinates of Pixel leading to the nearest node
7650  ! If Up=1 then Pixl above node, else Pixl below node
7660    INTEGER Ynn,Ysw,Ndb,Lyr
7670    LET Lup=BIT(Lup,0)
7680    LET Lyr=SHIFT(Lszn,1)             ! in essence divides by 2
7690    LET Ynn=1+((Yn-1) MOD Lszn)       ! wrap around in Y coordinate
7700    IF Ynn<1 THEN Ynn=Ynn+Lszn
7710    IF Xn>0 AND Xn<=Lszn THEN
7720      LET Ndb=1
7730      IF Xn=1 AND Lup THEN LET Ndb=0
7740      IF Xn=Lszn AND NOT (Lup) THEN LET Ndb=SHIFT(Lszn*Lszn,1)-Lyr+1
7750      IF Ndb=1 THEN                   ! executes if node not yet assigned
7760        LET Xsw=(BIT(Xn,0) EXOR Lup)! 0=even 1=odd in Xn
7770        LET Ndb=(Xn-1-Lup)*Lyr+SHIFT(Ynn+Xsw,1)! basic node count
7780        IF Ynn=1 AND NOT (Xsw) THEN LET Ndb=Ndb+Lyr
7790      END IF
7800    ELSE
7810      LET Ndb=-1
7820    END IF
7830    RETURN Ndb
7840  FNEND
7850  ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
7860  SUB Cvndi(INTEGER Nodi,Lszi,Sctri,Xouti,Youti)
7870  ! Converts a node number of layering scheme into the (x,y)
7880  ! coordinates of neighboring nodes
7890  ! IN   Nodi = node number
7900  ! IN   Lszi = Pixl extent along either X or Y
7910  ! IN   Sctri = selection adjacent Pixl neighbor to node
7920  !      (0= X lower,Y lower; 1= X lower,Y higher;
7930  !       2= X higher, Y lower; 3= X higher, Y higher)
7940  ! OUT Xouti = X coordinate address outcome
7950  ! OUT Youti = Y coordinate address outcome
7960  ! internal variables:
7970  !  Lyri = a counter for number of layers
7980  !  Swi = an even odd switch
7990  !  Ovri = overfill counter
8000  !  Nodmaxi = maximum node for given Pixel grid size, Lszi
```

```
8010 !  Hfszi = half of Pixel grid size, Lszi (then Lszi must be even)
8020   INTEGER Lyri,Swi,Ovri,Nodmaxi,Hfszi
8030   IF BIT(Lszi,0) THEN PRINT " warning, odd Pixel grid extent"
8040   LET Nodmaxi=SHIFT(Lszi*Lszi,1)-Lszi+2
8050   LET Hfszi=SHIFT(Lszi,1)          ! effectively DIV 2 operation
8060   SELECT Nodi
8070   CASE <1
8080     LET Xouti=0
8090     LET Youti=0
8100   CASE >=Nodmaxi
8110     LET Xouti=Lszi+1
8120     LET Youti=Xouti
8130   CASE ELSE
8140     LET Lyri=1+((Nodi-1) DIV (Lszi-1))    ! for Bilayer
8150     LET Ovri=1+((Nodi-1) MOD (Lszi-1))    ! for # Nodes in Bilayer
8160     LET Swi=(Ovri>Hfszi)                  ! 0=lower 1=upper in Bilayer
8170     IF Swi THEN LET Ovri=Ovri-Hfszi       ! adjust for # Nodes in upper
8180     LET Xouti=SHIFT(Lyri,-1)-1+Swi+BIT(Sctri,1)
8190     LET Youti=SHIFT(Ovri,-1)-1+Swi+BIT(Sctri,0)
8200   END SELECT
8210 SUBEND
8220 ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
8230 SUB Cvndp(INTEGER Nodp,Lszp,Sctrp,Xoutp,Youtp)
8240 ! Converts a node number of layering scheme into the (x,y)
8250 ! coordinates of neighboring nodes
8260 ! IN   Nodp = node number
8270 ! IN   Lszp = Pixl extent along either X or Y
8280 ! IN   Sctrp = selection adjacent Pixl neighbor to node
8290 !      (0= X lower,Y lower; 1= X lower,Y higher;
8300 !       2= X higher, Y lower; 3= X higher, Y higher)
8310 ! OUT Xoutp = X coordinate address outcome
8320 ! OUT Youtp = Y coordinate address outcome
8330 ! internal variables:
8340 !  Ovrp = overfill counter
8350 !  Nodmaxp = maximum node for given Pixel grid size, Lszp
8360 !  Hfszp = half of Pixel grid size, Lszp (then Lszp must be even)
8370   INTEGER Ovrp,Nodmaxp,Hfszp
8380   IF BIT(Lszp,0) THEN PRINT " warning, odd Pixel grid extent"
8390   LET Hfszp=SHIFT(Lszp,1)          ! effectively DIV 2 operation
8400   LET Nodmaxp=SHIFT(Lszp*Lszp,1)-Hfszp+1
8410   SELECT Nodp
8420   CASE <1
8430     LET Xoutp=0
8440     LET Youtp=0
8450   CASE >=Nodmaxp
8460     LET Xoutp=Lszp+1
8470     LET Youtp=Xoutp
8480   CASE ELSE
8490     LET Xoutp=1+((Nodp-1) DIV Hfszp)      ! for slab of X
8500     LET Ovrp=1+((Nodp-1) MOD Hfszp)       ! for # Nodes within slab
8510     LET Youtp=SHIFT(Ovrp,-1)+BIT(Sctrp,0)-BIT(Xoutp,0)
8520     LET Xoutp=Xoutp+BIT(Sctrp,1)
8530     IF Youtp>Lszp THEN Youtp=Youtp-Lszp
8540   END SELECT
```

```
8550 SUBEND
8560 ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
8570 DEF FNMatprnt(Matrx(*),INTEGER Ordr)
8580 !***> Printout of an array sized Ordr x Ordr & @ element Spcs wide
8590   COM /Pass/ Relay
8600   INTEGER Sbcx,Sbcy,Sbcyy,Sbc,Typ
8610   LET Sbc=SGN(Ordr)               ! Reverse printout of rows indicator
8620   LET Pslct=0                     ! Pslct is the biggest element magnitude
8630   FOR Sbcy=1 TO ABS(Ordr)
8640     FOR Sbcx=1 TO ABS(Ordr)
8650       IF Pslct<ABS(Matrx(Sbcx,Sbcx)) THEN LET Pslct=ABS(Matrx(Sbcx,Sbcx))
8660     NEXT Sbcx
8670   NEXT Sbcy
8680   IF Pslct>0 THEN LET Typ=2-INT(LGT(Pslct))
8690   IF Typ=4 OR Typ=5 THEN Typ=3
8700   IF Typ=-1 THEN Typ=0
8710   FOR Sbcy=1 TO ABS(Ordr)
8720     LET Sbcyy=Sbc*Sbcy+(ABS(Ordr)+1)*(1-Sbc)/2
8730     FOR Sbcx=1 TO ABS(Ordr)
8740       SELECT Typ
8750       CASE =0
8760         PRINT USING "DDDDD,#";PROUND(Matrx(Sbcx,Sbcyy),0)
8770       CASE =1
8780         PRINT USING "DDD.D,#";PROUND(Matrx(Sbcx,Sbcyy),-1)
8790       CASE =2
8800         PRINT USING "DD.DD,#";PROUND(Matrx(Sbcx,Sbcyy),-2)
8810       CASE =3
8820         PRINT USING "D.DDD,#";PROUND(Matrx(Sbcx,Sbcyy),-3)
8830       CASE ELSE
8840         LET Typ=SGN(Typ)*99
8850       END SELECT
8860     NEXT Sbcx
8870     IF ABS(Typ)<>99 THEN PRINT
8880   NEXT Sbcy
8890   IF Typ=99 THEN PRINT "  . . . array too small to format"
8900   IF Typ=-99 THEN PRINT "  . . . array too big to format"
8910   RETURN 1
8920 FNEND
8930 ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
8940 DEF FNWnr(Diel(*),Frpx(*),Din,Pcterr,INTEGER Nth)
8950 !***> Object of function to find alf satisfying  1=SUM of(vol*diel^alf)
8960 !***>  where vol=fractional volumes
8970 !***>        diel=(species permittivity)/(composite permittivity)
8980 !***>        alf= constant exponential between -1 & +1 (series<->parallel)
8990   COM /Pass/ Relay
9000   INTEGER Spk,Arnd
9010   IF Din=0 OR MAX(Frpx(*))>=1 THEN
9020     LET Alf1=0
9030     LET Pcterr=1.E-99
9040     PRINT " .. series <=> parallel factor at or beyond limits"
9050   ELSE
9060     LET Gsum=0
9070     LET Gdev=0
9080     FOR Spk=1 TO Nth
```

```
9090         LET Gratio=0
9100         IF Diel(Spk)>0 AND Frpx(Spk)>0 THEN
9110           LET Gratio=LOG(Diel(Spk)/Din)
9120           LET Gsum=Gsum+Frpx(Spk)*Gratio
9130           LET Gdev=Gdev+Frpx(Spk)*Gratio*Gratio
9140         END IF
9150       NEXT Spk
9160       LET Alf1=-2*Gsum/Gdev
9170       LET Gsum=Gsum+Gdev                    ! Initial guess of sum
9180       LET Try=1                             ! Optimize sign in iterations
9190       LET Alf0=0
9200       LET Arnd=1
9210       WHILE ABS(Alf1-Alf0)>.0000000001
9220         LET Gdev=0
9230         LET Gsum=0
9240         FOR Spk=1 TO Nth
9250           LET Gratio=Diel(Spk)/Din          ! relative permittivity to composite
9260           LET Gwk=0                         ! a term @ species to sum
9270           IF Gratio<>0 AND Frpx(Spk)>0 THEN
9280             LET Gwk=Frpx(Spk)*Gratio^Alf1
9290             LET Gsum=Gsum+Gwk               ! sum function (as described)
9300             LET Gdev=Gdev+LOG(Gratio)*Gwk   ! 1st derivative
9310           END IF
9320         NEXT Spk
9330         LET Pcterr=Gsum
9340         LET Gsum=Alf1*(Gsum-1)/Gdev
9350         LET Gdev=Alf1*Alf1-Try*Gsum
9360         IF Gdev<0 THEN Gdev=Gdev+2*Try*Gsum
9370         LET Alf2=SGN(Alf1)*SQR(Gdev)
9380         IF Arnd MOD 2=0 AND ABS(Alf2-Alf1)>ABS(Alf1-Alf0) THEN Try=-Try
9390         LET Alf0=Alf1                       ! bumping iterations
9400         LET Alf1=Alf2
9410         LET Arnd=Arnd+1                     ! incrementor
9420       END WHILE
9430       IF Abs(Alf1<.0000001) THEN
9440         LET Pcterr=100*(Pcterr-1)
9450       ELSE
9460         LET Pcterr=100*(Pcterr^(1/Alf1)-1)
9470       END IF
9480     END IF
9490     RETURN Alf1
9500   FNEND
9510   ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
9520   SUB Pixl2d_tilt
9530   ! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
9540   ! This program is for patterning a dielectric bond grid (rectangular
9550   ! type) with a binary mixture.  The interface is approximately flat
9560   ! at a tilted sngle.  (Say like a tilted glass of water)
9570   !                          +---------+
9580   !                          l..       l
9590   !                          l   ..    l
9600   !                          l       ..l
9610   !                          +---------+
9620   ! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
9630    DEG
9640    COM /Pass/ Relay
9650    COM /Pixel/ Chdr$[80],Dhdr$[80],INTEGER Lbonds,Pixl(1:180,1:180)
9660    INTEGER Accum,Opt,Ip,Jp
9670    REDIM Pixl(1:Lbonds,1:Lbonds)
9680    PRINT
9690    PRINT " Makes an bond grid pattern with a binary mixture whose ";
9700    PRINT "interface level sits at a selected volume fraction & tilt."
9710    INPUT "Request the desired volume percentage of the 1st component",Cmp
9720    INPUT "& request the desired tilt angle (in degrees)",Degtlt
9730    PRINT "The requested volume is";Cmp;"% & tilt angle is";Degtlt;"deg"
9740    PRINT "for a square grid of bonds of size";Lbonds;"X";Lbonds;"."
9750    DISP "Decide fate of straddlers ";
9760    INPUT "1) pixel pop/volume demand 2) individual volume sway",Optn
9770    LET Cmp=Cmp/100
9780    LET Pmc=1-Cmp
9790   ! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
9800   ! From the information of the two components & the angle of tilt
9810   ! then interfacial level can be drawn across the grid window.
9820   ! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
9830    LET Sn=ABS(SIN(Degtlt))           ! Sine
9840    LET Cs=ABS(COS(Degtlt))           ! Cosine
9850    LET Crit=MIN(Sn,Cs)/MAX(Sn,Cs)/2! Crictical volume transition
9860    Dgmin=MIN(Sn,Cs)/(Sn+Cs)          ! A transition on the diagonal
9870    LET Dgsn=Sn/(Sn+Cs)               ! Diagonal threshold at Crit vol
9880    LET Dgcs=1-Dgsn                   !  & its complement
9890    SELECT Cmp                        ! y-intercept for different cases
9900    CASE <=Crit
9910      LET Cpt=SQR(2*Cmp*Dgsn*Dgcs)
9920    CASE >=(1-Crit)
9930      LET Cpt=1-SQR(2*Pmc*Dgsn*Dgcs)
9940    CASE ELSE
9950      LET Cpt=(Cmp-.5)/(1+2*Crit)+.5
9960    END SELECT                        ! Done computing diagonal intercept
9970   !PRINT "The level has slope";PROUND(Sn/Cs,-2);"& a diag-intercept of";
9980   !PRINT PROUND(Cpt,-3);"."
9990   ! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
10000  !  The array will be filled according to whether the element
10010  !  happens to be on one or the other side of the tilt line.
10020  !  If an element is crossed through by the tilt line then
10030  !  the element will be assigned to the componet possessing
10040  !  the greatest volume deficet as the pixels are filled.
10050  ! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
10060   LET Vbias=0                              ! Initialize keep tabs of excess
10070   LET Accum=0                              ! Initialize totals counter
10080   FOR Ip=1 TO Lbonds
10090     FOR Jp=1 TO Lbonds
10100       Diag=Jp*Dgsn+Ip*Dgcs-.5              ! Diag intercept @pixel
10110       Dif=Diag-Cpt*Lbonds                  ! Distance offset on diag
10120       REM PRINT PROUND(Diag,-2);PROUND(Dif,-2);  ! Matrix calc out
10130       SELECT Dif
10140       CASE <=-.5
10150         Pixl(Ip,Jp)=1                      ! Assign 1st component
10160       CASE >=.5
```

```
10170           Pixl(Ip,Jp)=2                      ! Assign 2nd component
10180         CASE ELSE                            ! Borderline pixel case
10190           Dpf=.5-Dif                         ! Diagonal within pixel
10200           SELECT Dpf
10210           CASE <Dgmin
10220             Vpxl=Dpf*Dpf/Dgsn/Dgcs/2
10230           CASE >(1-Dgmin)
10240             Vpxl=1-((1-Dpf)^2)/Dgsn/Dgcs/2
10250           CASE ELSE
10260             Vpxl=(Dpf-.5)*(1+2*Crit)+.5
10270           END SELECT                         ! Vol frac within a pixel
10280           REM PRINT PROUND(Vpxl,-2);         ! For matrix varibles printout
10290           LET Vbias=Vbias+Vpxl               ! Accum of excess component 1
10300           IF Optn=2 THEN Vbias=Vpxl          ! Option of straightfoward bias
10310           IF Vbias>.5 THEN
10320             Pixl(Ip,Jp)=1
10330             Vbias=Vbias-1
10340           ELSE
10350             Pixl(Ip,Jp)=2
10360           END IF
10370         END SELECT
10380         Accum=Accum+2-Pixl(Ip,Jp)          ! Accumulation of 1st component
10390       NEXT Jp
10400     NEXT Ip
10410 SUBEND
10420 ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
10430 SUB Pixl2d_rand
10440 !***> Subprogram to create a 2D pixel array of random distribution
10450 !  @$%^%*^&%)!  R A N D O M   2 D   P I X E L   G R I D  &^$!$%)(%'$*
10460   COM /Pass/ Relay
10470   COM /Pixel/ Chdr$[80],Dhdr$[80],INTEGER Lpix,Pixl(1:180,1:180)
10480   INTEGER Xp,Yp,Xq,Yq,Fill,Frdm,Sqrs,Pxs,Pixtmp,When
10490   REDIM Pixl(1:Lpix,1:Lpix)
10500   PRINT " Enjoy creating a randomized 2D pixel grid whose pixel elements"
10510   PRINT " are labelled 1..9"
10520   INPUT "Random seed? (integer or neg if to be via timer)",Vchk
10530  !LET Vchk=Relay
10540   IF Vchk=0 THEN RANDOMIZE INT(TIMEDATE MOD 32767)
10550   IF Vchk>0 THEN RANDOMIZE INT(Vchk)
10560   LET Fill=0
10570   LET Sqrs=Lpix*Lpix
10580   LET Frdm=0
10590   LET When=0
10600   LET Pxs=0
10610   LET Yp=1
10620   LET Vchk=0
10630  !let Rq=100*Relay/64      ! if to program using the COM Relay
10640   FOR Fill=1 TO Sqrs
10650     IF Pxs<1 THEN
10660       LET When=When+1
10670       DISP "Filling with component [";VAL$(When);"], give volume ";
10680       INPUT "percent",Rq
10690       IF Rq<0 THEN
10700         LET Pxs=Sqrs-Fill+1
```

```
10710         ELSE
10720           LET Pxs=INT(Rq*Sqrs*.01+.5)
10730           IF Pxs>Sqrs-Fill+1 THEN LET Pxs=Sqrs-Fill+1
10740         END IF
10750         PRINT " Component [";VAL$(When);"] is assigned";Pxs;"pixels"
10760       END IF
10770       LET Xp=Fill-Frdm
10780       IF Xp>Lpix THEN
10790         LET Xp=Xp-Lpix
10800         LET Yp=Yp+1
10810         LET Frdm=Frdm+Lpix
10820       END IF
10830       IF Pxs>0 THEN LET Pixl(Xp,Yp)=When
10840       Pxs=Pxs-1
10850     NEXT Fill
10860     PRINT
10870 !***> lotto-ing or random mixing
10880     FOR Fill=1 TO Sqrs
10890       LET Frdm=INT(1+RND*Sqrs)
10900       IF Fill<>Frdm AND Frdm<=Sqrs THEN
10910         LET Xp=1+((Fill-1) MOD Lpix)
10920         LET Yp=1+((Fill-1) DIV Lpix)
10930         LET Xq=1+((Frdm-1) MOD Lpix)
10940         LET Yq=1+((Frdm-1) DIV Lpix)
10950         LET Pixtmp=Pixl(Yp,Xp)
10960         LET Pixl(Yp,Xp)=Pixl(Yq,Xq)
10970         LET Pixl(Yq,Xq)=Pixtmp
10980       END IF
10990     NEXT Fill
11000 SUBEND
11010 ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
11020 SUB Pixl2d_fill
11030 !***> Subprogram to output a pixel array hand filled by user
11040     COM /Pass/ Relay
11050     COM /Pixel/ Chdr$[80],Ohdr$[80],INTEGER Lpix,Pixl(1:180,1:180)
11060     INTEGER Xpix,Ypix,Fill,Xp,Yp
11070     REDIM Pixl(1:Lpix,1:Lpix)
11080     MAT Pixl= (0)
11090     FOR Xpix=1 TO Lpix
11100       FOR Ypix=1 TO Lpix
11110         CLEAR SCREEN
11120         PRINT
11130         FOR Xp=Lpix TO 1 STEP -1
11140           FOR Yp=1 TO Lpix
11150             PRINT USING "DD,#";Pixl(Xp,Yp)
11160           NEXT Yp
11170           PRINT
11180         NEXT Xp
11190         PRINT
11200         DISP "Filling at (";Xpix;",";Ypix;")";
11210         INPUT " specify species type with integer 1..9",Newpix
11220         LET Newpix=Newpix MOD 10
11230         LET Pixl(Xpix,Ypix)=Newpix
11240       NEXT Ypix
```

```
11250   NEXT Xpix
11260   CLEAR SCREEN
11270   FOR Xp=Lpix TO 1 STEP -1
11280     FOR Yp=1 TO Lpix
11290       PRINT USING "DD,#";Pixl(Xp,Yp)
11300     NEXT Yp
11310     PRINT
11320   NEXT Xp
11330 SUBEND
11340 ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
11350 SUB Pixl2d_ellps
11360 ! >  >  >  Subprogram to fill a pixel window with an ellipse  <  <  <
11370   COM /Pass/ Relay
11380   COM /Pixel/ Chdr$[80],Dhdr$[80],INTEGER Lwndw,Pixl(1:180,1:180)
11390   INTEGER Opt,Xp,Yp
11400   REDIM Pixl(1:Lwndw,1:Lwndw)
11410   PRINT "Choice of filling pixel grid with an ellipse"
11420   DISP "Volume percentage of [1]s to be filled by ellipse";
11430   INPUT " (if>.5 then host)",Vlm
11440   LET Vlm=Vlm/100
11450   DISP "Type 1 or 2 to opt for ";
11460   INPUT "1) eccentricty or 2) axis ratio",Opt
11470   DEG
11480   IF Opt=1 THEN
11490     INPUT "  Eccentricity?",Ecn
11500     IF ABS(Ecn)>1 THEN PRINT " Too eccentric"
11510     LET Mnjr=SQR(1-Ecn*Ecn)
11520   ELSE
11530     INPUT "  Ratio  (minor axis)/(major axis)?",Mnjr
11540     IF ABS(Mnjr)>1 THEN Mnjr=1/Mnjr
11550     LET Ecn=SQR(1-Mnjr*Mnjr)
11560   END IF
11570   INPUT " Angle of major axis w.r.t. horizontal in degrees?",Mang
11580   PRINT "The ellipse characteristics are:"
11590   PRINT " eccentricity=";PROUND(Ecn,-4);
11600   PRINT ", axis ratio=";PROUND(Mnjr,-4);", &"
11610   IF Vlm>.5 THEN LET Vlm=.5-Vlm
11620   LET Major=SQR(ABS(Vlm)/Mnjr/PI)
11630   LET Minor=Mnjr*Major
11640   PRINT " measures a sexy";PROUND(2*Minor,-4);
11650   PRINT "by";PROUND(2*Major,-4);"w.r.t fraction of pixel window."
11660   FOR Xp=1 TO Lwndw
11670     LET Xpos=((Xp-Lwndw DIV 2)-.5)/Lwndw
11680     IF ABS(Xpos)>.5 THEN PRINT " Warning.. touches window edge"
11690     LET Xcs=Xpos*COS(Mang)
11700     LET Xsn=Xpos*SIN(Mang)
11710     FOR Yp=1 TO Lwndw
11720       LET Ypos=((Yp-Lwndw DIV 2)-.5)/Lwndw
11730       IF ABS(Ypos)>.5 THEN PRINT " Warning.. touches window edge"
11740       LET Xpos=Xcs-Ypos*SIN(Mang)
11750       LET Ypos=Ypos*COS(Mang)+Xsn
11760       LET Gfct=(Ypos/Major)^2+(Xpos/Minor)^2
11770       IF Gfct>1 THEN Pixl(Xp,Yp)=2-(1-SGN(Vlm))/2
11780       IF Gfct<=1 THEN Pixl(Xp,Yp)=1+(1-SGN(Vlm))/2
```

```
11790     NEXT Yp
11800   NEXT Xp
11810 SUBEND
11820 ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
11830 SUB Pixl2d_strat
11840 ! Pixel gridding of stratified layers
11850   COM /Pass/ Relay                        ! for sharing to subs
11860   COM /Pixel/ Chdr$[80],Dhdr$[80],INTEGER Lxtnt,Pixl(1:180,1:180)
11870   INTEGER Strat,Stack,Fix,Lcnt,Is,Ns,Swap
11880   REDIM Pixl(1:Lxtnt,1:Lxtnt)
11890   INPUT "Layer sequencing? random=1 by user=2",Strat
11900   IF Strat<0 THEN STOP
11910   LET Strat=(Strat>1)                    ! reassigned 0 or 1
11920   IF NOT (Strat) THEN
11930     INPUT "Random seed? (give 0 via timer or give integer)",Seed
11940     IF Seed<0 THEN STOP
11950     LET Swap=INT(Seed+.5)
11960     IF Swap=0 THEN
11970       RANDOMIZE INT(TIMEDATE MOD 32767)
11980     ELSE
11990       RANDOMIZE Swap
12000     END IF
12010   END IF                                 ! Strat test RANDOMIZE
12020   DISP "Orientation of layers? flatwise=-1 spanned=+1 (or series<=>";
12030   INPUT "parallel)",Stack
12040   LET Stack=(Stack>=0)                    ! reassigned 0 or 1
12050   IF Strat THEN
12060     LET Is=1
12070     FOR Lcnt=1 TO Lxtnt
12080       LET Ns=0
12090       DISP "Assign to layer";Lcnt;"a component type [1..9]";
12100       INPUT " or null to default to last",Ns
12110       IF Ns<0 THEN STOP
12120       IF Ns=0 THEN                   ! defaults
12130         LET Ns=Is
12140       ELSE
12150         LET Is=Ns
12160       END IF
12170       PRINT " Layer";Lcnt;"assigned to component type [";VAL$(Ns);"]"
12180       LET Pixl(1,Lcnt)=Ns
12190     NEXT Lcnt
12200   ELSE
12210     LET Is=Lxtnt
12220     FOR Lcnt=1 TO 9
12230       DISP Is;"layers remain, how many? or what fraction?";
12240       DISP " belong with type [";VAL$(Lcnt);"]";
12250       INPUT "",How
12260       IF How<0 THEN LET Lcnt=10
12270       IF How>0 THEN
12280         IF How<1 THEN
12290           LET Ns=INT(How*Lxtnt+.5)
12300         ELSE
12310           LET Ns=INT(How+.5)
12320         END IF
```

```
12330          FOR Fix=Is TO Is-Ns STEP -1
12340            IF Fix>0 THEN LET Pixl(1,Fix)=Lcnt
12350          NEXT Fix
12360          LET Is=Is-Ns
12370        END IF                     ! How>0 test
12380        IF Is<1 THEN Lcnt=10
12390      NEXT Lcnt
12400      FOR Lcnt=1 TO Lxtnt
12410        LET Ns=1+INT(RND*Lxtnt)
12420        IF Lcnt<>Ns AND Ns<=Lxtnt THEN
12430          LET Swap=Pixl(1,Lcnt)
12440          LET Pixl(1,Lcnt)=Pixl(1,Ns)
12450          LET Pixl(1,Ns)=Swap
12460        END IF
12470      NEXT Lcnt
12480    END IF                           ! Strat test
12490    IF Stack THEN                    ! rest parallel stacking
12500      FOR Lcnt=1 TO Lxtnt
12510        FOR Fix=2 TO Lxtnt
12520          LET Pixl(Fix,Lcnt)=Pixl(1,Lcnt)
12530        NEXT Fix
12540      NEXT Lcnt
12550    ELSE                             ! rest by series stacking
12560      FOR Lcnt=Lxtnt TO 1 STEP -1
12570        LET Swap=Pixl(1,Lcnt)
12580        FOR Fix=Lxtnt TO 1 STEP -1
12590          LET Pixl(Lcnt,Fix)=Swap
12600        NEXT Fix
12610      NEXT Lcnt
12620    END IF                           ! Stack test
12630 SUBEND
12640 ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
12650 SUB Pixl2d_cbox
12660 ! subroutine to make concentric boxes
12670    COM /Pass/ Relay                     ! for sharing to subs
12680    COM /Pixel/ Chdr$[80],Dhdr$[80],INTEGER Lxtnt,Pixl(1:180,1:180)
12690    INTEGER Cntr,Shell,Fix,Lcnt,Peri,Cntrk,Past
12700    REDIM Pixl(1:Lxtnt,1:Lxtnt)
12710    INPUT "Start where? corner=0 center=1",Cntr
12720    IF Cntr<0 THEN STOP
12730    LET Cntr=BIT(Cntr,0)
12740    IF Cntr THEN
12750      LET Cntrk=SHIFT(Lxtnt,1)
12760    ELSE
12770      LET Cntrk=Lxtnt
12780    END IF
12790    LET Past=1                        ! default assignment
12800    FOR Shell=1 TO Cntrk
12810      LET Fix=0
12820      LET Peri=SHIFT(Shell,-1)-1        ! 1/4 perimeter=2*Shell-1
12830      DISP "At concentric box shell ";Shell;"containing";
12840      DISP SHIFT(Peri,-SHIFT(Cntr,-1));"pixels, assign";
12850      INPUT " component type [1,..9]",Fix
12860      IF Fix<0 THEN STOP
```

```
12870      IF Fix=0 THEN
12880        LET Fix=Past
12890      ELSE
12900        LET Past=Fix
12910      END IF
12920      PRINT "   Shell";Shell;"containing";SHIFT(Peri,-SHIFT(Cntr,-1));
12930      PRINT "pixels is assigned component type, [";VAL$(Fix);"]";" from";
12940      IF Cntr THEN PRINT " center."
12950      IF NOT (Cntr) THEN PRINT " corner."
12960      FOR Lcnt=1 TO SHIFT(Shell,-Cntr)
12970        IF Cntr THEN
12980          LET Pixl(Cntrk-Shell+1,Cntrk-Shell+Lcnt)=Fix
12990          LET Pixl(Cntrk+Shell,Cntrk+Shell-Lcnt+1)=Fix
13000          LET Pixl(Cntrk-Shell+Lcnt,Cntrk+Shell)=Fix
13010          LET Pixl(Cntrk+Shell-Lcnt+1,Cntrk-Shell+1)=Fix
13020        ELSE
13030          LET Pixl(Shell,Lcnt)=Fix
13040          LET Pixl(Lcnt,Shell)=Fix
13050        END IF
13060      NEXT Lcnt
13070    NEXT Shell
13080 SUBEND
13090 ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
13100 REM A subroutine to make a Pixel grid with an ellipse inclusion
13110 REM imbedded in a host.  The ellipse component is always the smaller
13120 REM of the two componnents and thus is symmetric with respect to the
13130 REM fractional volume filling factor.  Since this is limited to only
13140 REM two components then the volume of the 2nd is used as the variable.
13150 COM /Pixel/ Chdr$[80],Dhdr$[80],INTEGER Lxtnt,Pixl(1:180,1:180)
13160 INTEGER Aparm,Abdf,Gde,Pdf,Ptst,Prq,Sqrs,Swup,Swpxs,Xp,Yp
```

Appendix II

Plot Program for Numerical Solutions

```
10    !   &   &   &   &   &   &   &   &   &   &   &   &   &   &   &   &   &
20    !   "PLOT_NWK" is a program designed to input dielectric data from
30    !   network simulations and plot it.                    July 1990
40    COM /Memr/ Vctrs(1:505,1:5),Hdra$[80],Hdrb$[80],Sfil$[64],INTEGER Dmx,Dmy
50    INTEGER Kif,Xpt,Ypt,Plt,Xlmn,Ylmn,Rpt,Subj,Mach,Eff,Cpy
60    DIM Clb$[1],Hlb$[40],Xlb$[40],Ylb$[40]
70    LET Kif=0
80    CLEAR SCREEN
90    PRINT RPT$(" ",24);"A   G R A P H   I S   W O R T H"
100   PRINT RPT$(" ",24);"M A N Y   D A T A   P O I N T S"
110   PRINT RPT$(" ",27);DATE$(TIMEDATE);" at ";TIME$(TIMEDATE)
120   PRINT
130   !***> Data preparation
140   INPUT " Indicate data source: 0=internal via COM /Memr/ 1=file ",Kif
150   IF Kif<0 THEN STOP
160   IF Kif=1 THEN
170     INPUT " Name the data source""file""",Sfil$
180     PRINT " Data comes from""";Sfil$;""":"
190     ASSIGN @Sourc TO Sfil$;FORMAT OFF
200     ENTER @Sourc;Hdra$,Hdrb$,Dmx,Dmy
210     REDIM Vctrs(1:Dmx,1:Dmy)
220     ENTER @Sourc;Vctrs(*)
230     ASSIGN @Sourc TO *
240   ELSE
250     PRINT " Data internal:"
260     IF Dmx>0 AND Dmy>0 THEN REDIM Vctrs(1:Dmx,1:Dmy)
270   END IF
280   PRINT Hdra$
290   PRINT Hdrb$
300   PRINT " Data stored as array( 1 :";Dmx;", 1 :";Dmy;" )";
310   IF Dmx<1 OR Dmy<1 THEN PRINT " ???"
320   IF Dmx>0 AND Dmy>0 THEN PRINT
330   LET Kif=0
340   INPUT " Type: 0) to skip listing 1) to list array",Kif
350   IF Kif<0 THEN STOP
360   IF Kif=1 THEN
370     FOR Xpt=1 TO Dmx
380       FOR Ypt=1 TO Dmy
390         PRINT Vctrs(Xpt,Ypt),
400       NEXT Ypt
410       PRINT
420     NEXT Xpt
430     WAIT .5
440   END IF
```

```
450   LET Subj=0
460   INPUT "Plot manner? defined later=0 ser<->prll=1, diel=2 LOGS=3",Subj
470   IF Subj<0 THEN STOP
480   IF Subj=0 THEN LET Subj=1  ! Defaults to Series<=>Parallel plot
490   LET Xlmn=0
500   DISP "Enter column vector 1..";VAL$(Dmy);" for X axis ";
510   INPUT "data?",Xlmn
520   IF (Subj=1 OR Subj=2) AND Xlmn=0 THEN LET Xlmn=2  ! Default in ser<=>prll
530   LET Aaa=MAXREAL
540   LET Zzz=MINREAL
550   FOR Xpt=1 TO Dmx
560     LET Tst=Vctrs(Xpt,Xlmn)
570     IF Tst<Aaa THEN LET Aaa=Tst
580     IF Tst>Zzz THEN LET Zzz=Tst
590   NEXT Xpt
600   IF Subj=1 THEN
610     LET Xlow=0
620     LET Xhigh=1
630   ELSE
640     DISP "Minimum of X axis data is ";Aaa;", what shall be minimum for";
650     INPUT " plot?",Xlow
660     DISP "Maximum of X axis data is ";Zzz;", what shall be maximum for";
670     INPUT " plot?",Xhigh
680   END IF
690   IF Subj=3 THEN
700     LET Xlow=LGT(ABS(Xlow))
710     LET Xhigh=LGT(ABS(Xhigh))
720   END IF
730   LET Xspan=Xhigh-Xlow
740   LET Xavg=(Xhigh+Xlow)*.5
750   IF Subj=3 THEN PRINT " LOG ";
760   PRINT " X axis of plot to range from ";Xlow;"to ";Xhigh;"."
770   LET Ylmn=0
780   DISP "Enter column vector 1..";VAL$(Dmy);" for Y axis ";
790   INPUT "data?",Ylmn
800   IF Subj=1 AND Ylmn=0 THEN LET Ylmn=4  ! Default in ser<=>prll case
801   IF Subj=2 AND Ylmn=0 THEN LET Ylmn=3  ! Default to permittivity case
810   LET Aaa=MAXREAL
820   LET Zzz=MINREAL
830   FOR Xpt=1 TO Dmx
840     LET Tst=Vctrs(Xpt,Ylmn)
850     IF Tst<Aaa THEN LET Aaa=Tst
860     IF Tst>Zzz THEN LET Zzz=Tst
870   NEXT Xpt
880   IF Subj=1 THEN
890     LET Ylow=-1
900     LET Yhigh=1
910   ELSE
920     DISP "Minimum of Y axis data is ";Aaa;", what shall be minimum for";
930     INPUT " plot?",Ylow
940     DISP "Maximum of Y axis data is ";Zzz;", what shall be maximum for";
950     INPUT " plot?",Yhigh
960   END IF
970   IF Subj=3 THEN
```

```
980    LET Ylow=LGT(ABS(Ylow))
990    LET Yhigh=LGT(ABS(Yhigh))
1000  END IF
1010  LET Yspan=Yhigh-Ylow
1020  LET Yavg=(Yhigh+Ylow)*.5
1030  LET Eff=0
1040  IF Subj=1 THEN
1050    INPUT "Overlay an effective medium [EM] curve for 2D? no=0 yes=1",Eff
1060    IF Eff<0 THEN STOP
1070    LET Eff=BIT(Eff,0)
1080    IF Eff THEN INPUT "Enter permittivity ratio for EM curve",Rem
1090    IF Rem<0 THEN STOP
1100    IF Rem=0 THEN LET Eff=0
1110  END IF
1120  IF Subj=3 THEN PRINT " LOG ";
1130  PRINT " Y axis of plot to range from ";Ylow;"to ";Yhigh;"."
1140  SELECT Subj
1150  CASE =0
1160    INPUT "Title?",Hlb$
1170    IF Hlb$="" THEN Hlb$=Hdra$[1,40]
1180    INPUT "X axis label?",Xlb$
1190    INPUT "Y axis label?",Ylb$
1200  CASE =1
1210    LET Hlb$="RANDOM 2D NETWORKS"
1220    LET Xlb$="Volume fraction of the larger"
1230    LET Ylb$="Averaging Parameter"
1240  CASE =2
1250    LET Hlb$="RANDOM 2D NETWORKS"
1260    LET Xlb$="Volume fraction of the larger"
1270    LET Ylb$="Permittivity"
1280  CASE =3
1290    INPUT "LOG Title?",Hlb$
1300    IF Hlb$="" THEN Hlb$=Hdra$[1,40]
1310    INPUT "LOG X axis label?",Xlb$
1320    INPUT "LOG Y axis label?",Ylb$
1330  END SELECT
1340  INPUT " Plot data 0) by points 1) by lines",Plt
1350  IF Plt<0 THEN STOP
1360  IF Plt=0 THEN
1370    INPUT " Type a single letter character for the plot points",Clb$
1380    INPUT " Character size in # of %s of the graph width? ie 1,2,..",Csz
1390    IF Csz=0 THEN LET Csz=1        ! Defaualt
1400  END IF
1410  IF Mach=0 THEN
1420    INPUT "Hardcopy of plot? 0=none 1=to printer 2=to plotter",Mach
1430    IF Mach<0 THEN STOP
1440  END IF
1450  IF Mach=2 THEN
1460    INPUT "Max speed of plotter pen? (~1-20, in cm/s)",Speed
1470    IF Speed<0 THEN Stop
1480    IF Speed=0 THEN LET Speed=10
1490  END IF
1500  WAIT 1
1510  CLEAR SCREEN
```

```
1520  KEY LABELS OFF
1530  !***> Begin plotting
1540  GINIT
1550  PLOTTER IS CRT,"INTERNAL"
1560  GRAPHICS ON
1570  FOR Cpy=0 TO (Mach>0)
1580   !VIEWPORT 20,120,15,85                    ! NEAR FULL SIZE SCREEN
1590    IF Subj=3 THEN
1600      VIEWPORT 40,95,35,90
1610    ELSE
1620      VIEWPORT 30,110,35,90
1630    END IF
1640    WINDOW Xlow,Xhigh,Ylow,Yhigh
1650    IF Plt=0 THEN
1660      FOR Xpt=1 TO Dmx
1670        LET Xx=Vctrs(Xpt,Xlmn)
1680       !IF Subj=1 THEN Xx=1-Xx               ! IF COMPLEMENTING NEEDED
1690        LET Yy=Vctrs(Xpt,Ylmn)
1700        IF Subj=3 THEN
1710          LET Xx=LGT(ABS(Xx))
1720          LET Yy=LGT(ABS(Yy))
1730        END IF
1740        MOVE Xx,Yy
1750        CSIZE Csz,.5
1760        LORG 5
1770        IF Clb$="" THEN
1780          LET Clb$="+"
1790          IF Subj<>1 AND Dmx<100 THEN LET Clb$=VAL$(Xpt)
1800        END IF
1810        LABEL Clb$
1820      NEXT Xpt
1830    ELSE
1840    !***> curve higher by a std deviation
1850 !  FOR Xpt=2 TO Dmx-1
1860 !    LET Xx=Vctrs(Xpt,1)
1870 !    LET Yy=Vctrs(Xpt,3)+Vctrs(Xpt,6)
1880 !    LINE TYPE 3
1890 !    PLOT Xx,Yy
1900 !  NEXT Xpt
1910 !  PENUP
1920    !***> central curve
1930      FOR Xpt=1 TO Dmx
1940        LET Xx=Vctrs(Xpt,Xlmn)
1950       !IF Subj=1 THEN Xx=1-Xx               ! IF COMPLEMENTING NEEDED
1960        LET Yy=Vctrs(Xpt,Ylmn)
1970        IF Subj=3 THEN
1980          LET Xx=LGT(ABS(Xx))
1990          LET Yy=LGT(ABS(Yy))
2000        END IF
2010        LINE TYPE 1
2020        PLOT Xx,Yy
2030      NEXT Xpt
2040      PENUP
2050    END IF
```

```
2060    IF Eff THEN                           ! overlay effective medium curve
2070      FOR Xpt=1 TO 200
2080        LET Xx=Xpt/201
2090        Aaa=(Rem-1)*(2*Xx-1)
2100        LET Dyf=.5*(Aaa+SQR(Aaa*Aaa+4*Rem))
2110        LET Yy=FNWnr(Rem,Xx,Dyf,Zzz)
2120        IF ABS(Zzz)>.000001 THEN PRINT " Warning! Ser<=>Prll difficulties"
2130        LINE TYPE 1
2140        PLOT Xx,Yy
2150      NEXT Xpt
2160    END IF
2170    IF Subj=1 THEN
2180      AXES .05,.1,0,-1,2,1
2190      AXES .05,.1,1,1,2,1
2200    ELSE
2210      FRAME
2220    END IF
2230    !***> set labels
2240    CLIP OFF
2250    LORG 4
2260    MOVE Xavg,Yhigh+.08*Yspan
2270    CSIZE 4,.6
2280    LABEL Hlb$
2290    LORG 4
2300    MOVE Xavg,Ylow-.16*Yspan
2310    CSIZE 4,.6
2320    LABEL Xlb$
2330    MOVE Xlow-(.1+(Subj=1)*.04)*Xspan,Yavg
2340    DEG
2350    LDIR 90
2360    LORG 4
2370    LABEL Ylb$
2380    IF Subj=1 THEN
2390      CSIZE 2.4,.6
2400      LORG 1
2410      MOVE Xlow-.092*Xspan,Ylow
2420      LABEL "Series-like"
2430      LORG 7
2440      MOVE Xlow-.092*Xspan,Yhigh
2450      LABEL "Parallel-like"
2460      LDIR 0
2470      LORG 5
2480      CSIZE 2.2,.6
2490      LET Zzz=Ylow-.04*Yspan
2500      FOR Xgf=Xlow TO Xhigh STEP .1
2510        MOVE Xgf-.01,Zzz
2520        LABEL USING "D.D";Xgf
2530      NEXT Xgf
2540      LET Aaa=Xlow-.05*Xspan
2550      FOR Ygf=Ylow TO Yhigh STEP .2
2560        MOVE Aaa,Ygf+.01
2570       !LDIR 90
2580        LABEL USING "DD.D";Ygf
2590      NEXT Ygf
```

```
2600     END IF
2610     LORG 1
2620     LDIR 0
2630     MOVE Xlow-.19*Xspan,Ylow-.30*Yspan
2640     CSIZE 2,.6
2650     LABEL Hdra$
2660     MOVE Xlow-.19*Xspan,Ylow-.35*Yspan
2670     LABEL Hdrb$
2680     SELECT Mach
2690     CASE =1
2700       DUMP GRAPHICS CRT
2710       WAIT 2
2720       LET Cpy=2
2730     CASE =2
2740       GINIT
2750       PLOTTER IS 705,"HPGL"
2760       GSEND "VS"&VAL$(Speed)
2770     END SELECT
2780   NEXT Cpy
2790   KEY LABELS ON
2800   PEN 0
2810   END
2820   DEF FNWnr(Dratio,Frpx,Din,Pcterr)
2830   !***> Object of function to find alf satisfying  1=SUM of(vol*diel^alf)
2840   !***>  where vol=fractional volumes
2850   !***>        diel=(species permittivity)/(composite permittivity)
2860   !***>        alf= constant exponential between -1 & +1 (series<->parallel)
2870     INTEGER Spk,Arnd
2880     IF Din=0 OR Frpx>=1 THEN
2890       LET Alf1=0
2900       LET Pcterr=1.E-99
2910       PRINT " .. series <=> parallel factor at or beyond limits"
2920     ELSE
2930       IF Dratio>0 AND Frpx>0 AND Frpx<1 THEN
2940         LET Gratio=LOG(1/Din)
2950         LET Gsum=(1-Frpx)*Gratio
2960         LET Gdev=(1-Frpx)*Gratio*Gratio
2970         LET Gratio=LOG(Dratio/Din)
2980         LET Gsum=Gsum+Frpx*Gratio
2990         LET Gdev=Gdev+Frpx*Gratio*Gratio
3000       END IF
3010       LET Alf1=-2*Gsum/Gdev
3020       LET Gsum=Gsum+Gdev                 ! Initial guess of sum
3030       LET Try=1                          ! Optimize sign in iterations
3040       LET Alf0=0
3050       LET Arnd=1
3060       WHILE ABS(Alf1-Alf0)>.0000000001
3070         LET Gsum=0
3080         LET Gdev=0
3090         LET Gratio=1/Din                 ! relative permittivity to composite
3100         LET Gwk=0                        ! a term @ species to sum
3110         IF Gratio<>0 AND Frpx>0 AND Frpx<1 THEN
3120           LET Gwk=(1-Frpx)*Gratio^Alf1
3130           LET Gsum=Gwk                   ! sum function (as described)
```

```
3140          LET Gdev=LOG(Gratio)*Gwk      ! 1st derivative
3150        END IF
3160        LET Gratio=Dratio/Din           ! relative permittivity to composite
3170        LET Gwk=0                       ! a term @ species to sum
3180        IF Gratio<>0 AND Frpx>0 AND Frpx<1 THEN
3190          LET Gwk=Frpx*Gratio^Alf1
3200          LET Gsum=Gsum+Gwk             ! sum function (as described)
3210          LET Gdev=Gdev+LOG(Gratio)*Gwk ! 1st derivative
3220        END IF
3230        LET Pcterr=Gsum
3240        LET Gsum=Alf1*(Gsum-1)/Gdev
3250        LET Gdev=Alf1*Alf1-Try*Gsum
3260        IF Gdev<0 THEN Gdev=Gdev+2*Try*Gsum
3270        LET Alf2=SGN(Alf1)*SQR(Gdev)
3280        IF Arnd MOD 2=0 AND ABS(Alf2-Alf1)>ABS(Alf1-Alf0) THEN Try=-Try
3290        LET Alf0=Alf1                   ! bumping iterations
3300        LET Alf1=Alf2
3310        LET Arnd=Arnd+1                 ! incrementor
3320      END WHILE
3330      IF Alf1<>0 AND Pcterr<>0 THEN
3340        LET Pcterr=100*(ABS(Pcterr)^(1/Alf1)-1)
3350      END IF
3360    END IF
3370    RETURN Alf1
3380  FNEND
3390  ! ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][
```

ELECTRONICS TECHNOLOGY AND DEVICES LABORATORY
MANDATORY DISTRIBUTION LIST
CONTRACT OR IN-HOUSE TECHNICAL REPORTS

101 Defense Technical Information Center*
ATTN: DTIC-FDAC
Cameron Station (Bldg 5)
Alexandria, VA 22304-6145

(*Note: Two copies for DTIC will be sent from STINFO Office.)

483 Director
US Army Material Systems Analysis Actv
ATTN: DRXSY-MP
001 Aberdeen Proving Ground, MD 21005

563 Commander, AMC
ATTN: AMCDE-SC
5001 Eisenhower Ave.
001 Alexandria, VA 22333-0001

609 Commander, LABCOM
ATTN: AMSLC-CG, CD, CS (In turn)
2800 Powder Mill Road
001 Adelphi, Md 20783-1145

612 Commander, LABCOM
ATTN: AMSLC-CT
2800 Powder Mill Road
001 Adelphi, MD 20783-1145

680 Commander,
US Army Laboratory Command
Fort Monmouth, NJ 07703-5000
1 - SLCET-DD
2 - SLCET-DT (M. Howard)
1 - SLCET-DB
35 - Originating Office

681 Commander, CECOM
R&D Technical Library
Fort Monmouth, NJ 07703-5000
1 - ASQNC-ELC-I-T (Tech Library)
3 - ASQNC-ELC-I-T (STINFO)

705 Advisory Group on Electron Devices
201 Varick Street, 9th Floor
002 New York, NY 10014-4877

ELECTRONICS TECHNOLOGY AND DEVICES LABORATORY
SUPPLEMENTAL CONTRACT DISTRIBUTION LIST
(ELECTIVE)

205 Director
Naval Research Laboratory
ATTN: CODE 2627
001 Washington, DC 20375-5000

221 Cdr, PM JTFUSION
ATTN: JTF
1500 Planning Research Drive
001 McLean, VA 22102

301 Rome Air Development Center
ATTN: Documents Library (TILD)
001 Griffiss AFB, NY 13441

437 Deputy for Science & Technology
Office, Asst Sec Army (R&D)
001 Washington, DC 20310

438 HQDA (DAMA-ARZ-D/Dr. F.D. Verderame)
001 Washington, DC 20310

520 Dir, Electronic Warfare/Reconnaissance
Surveillance and Target Acquisition Ctr
ATTN: AMSEL-EW-D
001 Fort Monmouth, NJ 07703-5000

523 Dir, Reconnaissance Surveillance and
Target Acquisition Systems Directorate
ATTN: AMSEL-EW-DR
001 Fort Monmouth, NJ 07703-5000

524 Cdr, Marine Corps Liaison Office
ATTN: AMSEL-LN-MC
001 Fort Monmouth, NJ 07703-5000

564 Dir, US Army Signals Warfare Ctr
ATTN: AMSEL-SW-OS
Vint Hill Farms Station
001 Warrenton, VA 22186-5100

602 Dir, Night Vision & Electro-Optics Ctr
CECOM
ATTN: AMSEL-NV-D
001 Fort Belvoir, VA 22060-5677

603 Cdr, Atmospheric Sciences Lab
LABCOM
ATTN: SLCAS-SY-S
001 White Sands Missile Range, NM 88002

607 Cdr, Harry Diamond Laboratories
ATTN: SLCHD-CO, TD (In turn)
2800 Powder Mill Road
001 Adelphi, MD 20783-1145